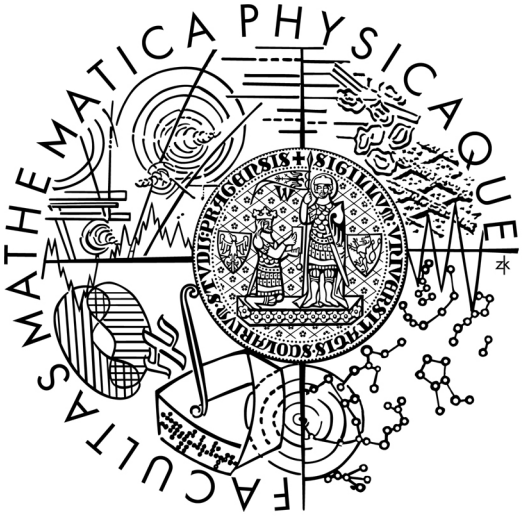
Charles University in Prague
Faculty of Mathematics and Physics

# DIPLOMA THESIS




**Rastislav Hudák**

# Node Alias Resolution

Department of Software Engineering
Supervisor: **RNDr. Leo Galamboš, Ph.D.**
Study program: **Computer Science, Software systems**

## Declaration

### Official version (Czech)

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

### Translated version (English)

I declare that I wrote my diploma thesis independently and exclusively with the use of the cited sources. I agree with lending the thesis.

Prague, 21th July 2009 Rastislav Hudák

# Table of contents

**Table of contents**

# Abstract

**Název práce:** Node Alias Resolution
**Autor:** Rastislav Hudák
**Katedra (ústav):** Katedra softwarového inženýrství
**Vedoucí diplomové práce:** RNDr. Leo Galamboš, Ph.D.
**e-mail vedoucího:** leo.galambos@mff.cuni.cz
**Abstrakt:** Při mapování počítačových sítí je důležité určit, které ze zjištěných IP adres patří příslušnému zařízení (směrovači). Směrovače mají zpravidla několik rozhraní. Zejména v případě, kdy zařízení odděluje dvě sítě z různých administrativních domén, bývají rozhraní směrovače pojmenovány (DNS jmény) různě a mají také přiděleny IP adresy z různých sítí. Práce se zabývá tímto problémem mapování IP adres na konkrétní směrovače, jenž se označuje jako "IP Alias Resolution". Problém lze zobecnit na shlukování IP adres do větších celků (například IP adresy jednoho autonomního systému nebo IP adresy v určité lokalitě), a proto má práce obecnejší název "Node Alias Resolution".
**Klíčová slova:** IP Alias Resolution, Node Alias Resolution, Mapování sítě

**Title:** Node Alias Resolution
**Author:** Rastislav Hudák
**Department:** Department of Software Engineering
**Supervisor:** RNDr. Leo Galamboš, Ph.D.
**Supervisor's e-mail address:** leo.galambos@mff.cuni.cz
**Abstract:** IP alias resolution is a common problem of all Internet mapping efforts based on traceroute tool. Routers often utilize multiple interfaces. When such a router is placed on administration boundaries, these interfaces can have non-adjacent IP addresses and DNS names. The challenge of this thesis is to find out which of the revealed router interfaces are aliases, meaning they belong to the same router. Because grouping of interfaces can have various granularity (router, ISP, geographical areas, etc.), we use more general term "Node Alias Resolution".
**Keywords:** IP alias resolution, Node alias resolution, Internet topology inferring

# 1 Introduction

**Summary.** A brief introduction to the IP alias resolution problem and this thesis.

**What is alias resolution.** Alias resolution, also known as IP alias resolution or node alias resolution is a process of resolving which interfaces belongs to a particular node in the topology graph of a computer network. Because alias resolution is not supported by any network protocol, used methods are only heuristics.

By far the most usual level at which computer network topologies are studied is OSI layer 3 level, usually referred as router level because of routers (devices operating at layer 3) being the devices that mostly interconnects networks.

Hence if a router level topology is desired, the aim of the alias resolution process will be to find out which interfaces (usually observed by traceroute tool), represented by their IP addresses, belongs to a single particular router. That is why this type of alias resolution is often referred as IP alias resolution.

If studying computer networks at another level, it is desired to obtain a different view of the network topology. An example may be a geographical topology, where a node may represent a POP location of an ISP, or an ISP level topology, where a node may represent whole single ISP. Processes leading to such topologies may also be referred to as alias resolution techniques.

However, router level topology, being the most precise computer network topology we can get by measuring via standard network protocols, is the most often desired, most often studied and probably the most interesting topology.

Therefore the IP alias resolution is exhaustively studied by various research groups, and therefore this thesis is also focused on IP alias resolution.

**Why IP alias resolution is needed.** When obtaining router level network topology, usually the traceroute tool is used for observing edges (interfaces or IP addresses) and vertices (links) of the desired graph.

If the measurement infrastructure contains single vantage point from where the traceroute tool observes the network, theoretically no alias resolution is

needed. However, due to routing policies used in Internet, it is impossible to obtain reasonable network topology with this approach as shown by Teixeira *et al.* in [1].

As soon as multiple vantage points are used[1] to obtain the set of links and interfaces, it is not possible to infer the topology from obtained dataset without additional analyses. This is due to the limitations of the traceroute tool and the protocol it is build on, as described later in this thesis.

Among others, a critical step in analyzing the traceroute dataset is the IP alias resolution. Without it, the resulting topology will contain too many nodes and links, and will be of little or no use at all as shown by Gunes and Sarac in [2] and [3], or by Willinger *et. al* [4].

**Why improved IP alias resolution techniques are needed.** While it is more that 10 years since the first alias resolution methods where developed, still the most state of the art techniques of today have significant drawbacks.

In practice this leads to approaches that combines known methods, while original methods are being improved and even new are being developed in an effort to reach as accurate network topologies as possible. Yet no method or approach is reliable enough to provide a network topology that match the reality[2].

While the accuracy of the alias resolution method is crucial, it is not the only property that is evaluated. There are others, like the amount of probes used if actively measuring, obtrusiveness of measurement or how much time it takes to provide reliable results.

Therefore, the research in this area is still actual and dynamic.

**The content of this thesis.** First, an analysis of several key terms used and referred by the alias resolution methods is provided. Second, the basic methods and techniques are explained. Finally original contribution of this thesis is presented, its theoretical basis, practical implementation and evaluation.

[1] Or other techniques, like loose source routing, are used to simulate more vantage points

[2] At least in cases where the real underlying network topology was known when evaluating performance of described methods.

# 1 Key terms

**Summary.** In this chapter several terms critical for alias resolution will be discussed in detail to provide a foundation for following chapters. Discussed terms may not seem to be closely related to alias resolution but are crucial in understanding the limitations and shortcomings of various alias resolution techniques.

## 1.1 UDP and TCP probes

**Probing.** Throughout this thesis, by a probe, a measurement packet will be understood. Its purpose is soliciting a request which will eventually provide desired information.

**TCP probes**. Probing in alias resolution methods is usually aimed to routers. Because routers are meant to work purely on Network Layer and TCP is a Transport Layer protocol, it is not possible to start and maintain a TCP connection with a router. Although most routers are able to do so (provides services running even on Application Layer), due to security reasons it is restricted for administration use only.

Therefore if sending TCP probe it is usually just TCP SYN packet, soliciting TCP RST packet in response (e.g. used by Bender *et al.* in [5]), as hosts may be configured to send TCP RST packets if TCP communication is blocked by firewall. An TCP RST packet is a full TCP packet and it may contain valuable information.

**UDP probes.** UDP packets are significantly smaller than TCP packets, thus UDP probes are used more often in measurement, unless hosts are not less responsive (or some TCP header field value is needed). Quite surprisingly, Bender *et al.* [5] in a recent study claimed routers to be more responsive to TCP probes.

Comparison of TCP, UDP and ICMP probe responsiveness is discussed in the last chapter of this thesis.

## 1.2 ICMP, TTL and Ping

**ICMP.** ICMP [6] is an Layer 3 network protocol, widely used in Internet, defining control and error messages. It is commonly used by various tools to reveal some properties of computer networks, such as delays between end hosts or the topology of a network.

**Messages.** ICMP messages, either requests or replies, are encapsulated in a single packet.

The header of the ICMP packet has following structure:

| Bits | 160-167 | 168-175 | 176-183 | 184-191 |
|---|---|---|---|---|
| **160** | Type | Code | Checksum | |
| **192** | ID | | Sequence | |

**Figure 1:** ICMP header

Types of ICMP messages are denoted by the Type field in the header of each ICMP packet. The Code field may denote further message type subdivision. Four important messages those will often be mentioned throughout the thesis are:

- Echo Request
- Echo Reply
- Destination Unreachable
- Time Exceeded

**Echo Request and Reply.** These two messages are at the heart of the ping tool. According to RFC 1122 these messages should always be processed:

> Every host MUST implement an ICMP Echo server function that receives Echo Requests and sends corresponding Echo Replies. [6]

However, due to frequent DoS attacks (e.g. Smurf attack [7]) leveraging from this convention, echo messages are often blocked on the end hosts by their firewalls. This turned out to have unfortunate consequences for tracert tool on Windows family operating systems, because Echo Request message is sent on

the last hop. Unix/Linux type operation systems uses UDP packet sent to 33434 or 33534 port for the last hop of their traceroute tool (soliciting Destination Port Unreachable message), therefore the lack of Echo messages support is not a serious concern.

Fortunately, probably because of protocol's crucial part in troubleshooting computer networks, important ICMP packets are usually not being ignored by routers (as opposed to end hosts). In experiments run by Burch in 2002 [8], only 66% of 587 828 IP addresses where responsive to UDP packets, while 92% were responsive to ICMP Echo Requests.

**Destination Unreachable.** This message type has many subtypes. Depending on the Code field of the header, the Destination Unreachable message bears various meanings:

- Destination Network Unreachable
- Destination Host Unreachable
- Destination Port Unreachable
- Destination Host Unknown
- etc.

In some alias resolution techniques, the Destination Port Unreachable message is used. An example is a technique used by Pansiot and Grad [9] referred as Source Address method in this thesis.

The method exploits following behavior specified in RFC 1812:

> Except where this document specifies otherwise, the IP source address in an ICMP message originated by the router MUST be one of the IP addresses associated with the physical interface over which the ICMP message is transmitted. [10]

This can not be achieved by Echo type messages, as in the ICMP protocol specification there is explicitly stated that:

> The IP source address in an ICMP Echo Reply MUST be the same as the specific-destination address ... of the corresponding ICMP Echo Request message. [6]

In fact, in experiments run by Barford *et al.* [11] a suspicion was raised that not all routers conforms the rule for returning ICMP messages (i.e. the Destination

Port Unreachable message used the destination address as its source address, instead of the address of outgoing interface).

**TTL.** To prevent packets for eventually looping forever in network, each IP packet has an 8-bit TTL (Time To Live) field in the header. When creating original packet to send, this field is set to some value from interval (0,255>. As stated in RFC 1812:

> When a router forwards a packet, it MUST reduce the TTL by at least one. If it holds a packet for more than one second, it MAY decrement the TTL by one for each second. [10]

The second rule is introduced due to the fact that initially TTL was meant to be a second counter. However, nowadays a hop lasts for less than a second and TTL is considered to have pure hop count meaning and implementation. Still, care have to be taken when relying on the assumption that each hop represents a single point-to-point link as some devices, due to this rule or due to misconfiguration or malfunction, are not handling TTL in the correct manner.

The protocol does not specify the initial TTL values, it is left for the implementation of the network stacks of specific operating systems. Therefore initial TTL[3] is considered useful for passive fingerprinting. It is also used to establish the hop distance of the remote host (this is used in some alias resolution methods). Unfortunately there are only few values used as initial TTL (common are: 30, 32, 60, 64, 128, 150, 255), with most common, by factor of almost 4 [8], being the 255.

**Time Exceeded message.** As a router receives a packet, it checks, whether the router itself is the packet's destination. If not, it decrements TTL. If TTL is 0 after decrementing, the packet is discarded and Time Exceeded message is generated and sent back to source. If TTL is not 0 packet is sent further to the network.

**Rate limit.** Due to attacks mentioned above, some ICMP messages are ignored and some are rate limited. For example, a common restrictions are to

[3] It can be inferred from the TTL value of the packets returned from remote host.

not reply to Address Mask Request message or to send only one Destination Unreachable message per second.

## 1.3 Traceroute

**Description.** The purpose of the traceroute tool is to discover routers residing between source (host issuing traceroute) and destination (target of traceroute). It does so by sending UDP datagrams or Echo requests to the destination, while incrementing initial TTL from 1 continuously until the destination is reached. First message, having TTL set to 1, only reaches first router. There it is discarded (as after decrementing the TTL will be 0) and Time Exceeded message is sent back to source. The traceroute tool at the source host then inspects source address in the Time Exceeded packet and presents it as the first hop (or router) on the way to destination. Then it sends second message, with initial TTL set to 2, obtaining the second hop and so on. In some implementations, the traceroute tool sends several probes in parallel to speed up the process.

**Traceroute's drawbacks.** Although traceroute is widely used as a main topology discovery tool (Skitter [12], Scamper [13], Rocketfuel [14], iPlane [15], Mercator [16]) it has some significant pitfalls [17]. Mainly:

- Probes blocked by firewalls.
- Load balancing may cause incorrect traceroute output.
- Presence of anonymous routers in output.
- Sampling bias if used as topology discovery tool.

This led to a development of many versions of traceroute tool (LBL traceroute, tracert, Paris traceroute, tcptraceroute, Paratraceroute, etc.).

Lakhina *et. al* [18], presented a paper showing the traceroute-based approach introduces a significant bias to the measurement. They argue that the Internet can not be modeled as a power-law random graph, although it may seem so from collected traceroutes[4].

[4] In such a graph, the degree distribution of nodes follows a distribution with a power-law tail.

**Avoiding firewalls.** One of the differences is the way in which the destination is contacted (or how the last hop in traceroute output is obtained). As stated, the last hop of traceroute is usually measured with sending UDP packets to some high numbered port (where no application is likely to listening) or Echo requests. In fact there are many more methods. For instance a TCP SYN packet can be sent to a port where applications tends to be running (like port 80). This techniques are used no only to avoid firewalls, but also to support better reply-to-request packet matching [13].

**Avoiding load balancing.** As for alias resolution traceroute is a prerequisite, not an integral part, details of various implementations of traceroute tools will not be covered in this thesis. Nevertheless the load balancing problem have to be emphasized, as it may introduce serious anomalies to measurement and so may cause errors in alias resolution methods if not taken into account.

To match an UDP packet request with its reply, often the port number is used, because UDP is a connectionless protocol and so nowhere is stated that the source address of the reply will be the same as the destination address of the request (this happens often, as observed by Burch in [8]). Each packet in traceroute is then sent with another port number. This has a negative consequence, that the checksums of these packets, computed by router's load balancer, will not be the same. Therefore the load balancer may sent packets belonging to the same traceroute by various links.

When using traceroute output, it is usually assumed[5] that the sequence of hops constitutes a true sequence of routers and links between them, or a real path the packet took to destination. This is only true if all packets of the traceroute were identically routed. As load balancer may not follow this expectation, some routers that are one hop from each other in the traceroute's output may not actually be connected by single link in the real topology.

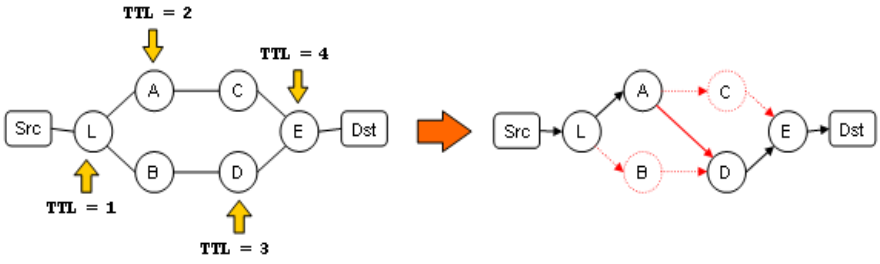



**Figure 2:** Example of how an incorrect traceroute output can be produced if load balancing is in use at the node L.

Paris traceroute [19] is a tool that avoids this by using other techniques for reply-to-request packet matching, leaving the checksummed part of the packet unchanged. Still, if a per packet load balancing is used this will not avoid erroneous traceroutes.

**Anonymous routers.** Anonymous routers, referred as *.*.*.* in traceroute output, are a serious problem for topology inferring based on traceroute.

Consider, that such routers can not be probed (as they are unresponsive) and can not be distinguished among other anonymous routers in other traceroutes. Therefore if a topology graph is to be the result of the measurement, each instance of anonymous router have to be a separate verticle. This may lead to a topology far from reality.

Unfortunately, it is not easy to deal with this problem. In a paper by Yao, *et al.* [20], a sophisticated technique is provided, while they show it is an NP-complete problem. An easy but not reliable approach is to use bisimilarity, meaning that each anonymous router, that has the same predecessor and successor among all obtained traceroutes, is considered the same one and so is represented by a single verticle in the topology graph. The disadvantage is, that with this approach each anonymous router in the resulting topology will have exactly two neighbors. This may often be a false representation.

However this approach may be sufficient. It was used by Bilir et al. [21] during topology measurement. If two consecutive unresponsive routers were observed, these were clustered and bisimilarity was used upon such a cluster. If more than two consecutive unresponsive routers were observed, the trace was discarded.

**Topology sampling algorithms.** Last problem that should be mentioned is the algorithm used for issuing traceroutes when discovering network topology. At first it may seem to be sufficient to use the naïve algorithm and to start collecting traceroutes to chosen IP addresses.

But first, IP addresses to trace have to be chosen. There are several ways:

- Addresses of web servers.
- Using algorithm for inferring IP addresses (e.g. starting from local network).

- Using BGP feeds to obtain existing networks.
- Other.

If a naïve algorithm will be used on a Internet scale, soon critical issues will arise:

- Sampling takes too long. No topology map that is obtained in more than a couple of days may be consistent, as Internet topology may change fast.
- Extensive sampling triggers IDS alarms in end networks.
- Many traces takes the same path (providing no new information).
- Most of the traces goes through the same routers at first hops (may be evaluated as DoS attack).

Considering this problems, building a solid sampling platform is a challenging task.

Interesting algorithms have been developed to avoid these issues. For instance Tangmunarunk *et. al* [16] developed a method called Informed Random Address Probing to guess addressable subnets (while developing the Mercator tool). Donnet *et. al* [22] introduced Doubletree algorithm, where:

> The key ideas are to exploit the treelike structure of routes to and from a single point in order to guide when to stop probing, and to probe each path by starting near its midpoint.

Also Zeitoun and Jamin [23] shown an algorithm to rapidly discover responsive networks. Spring *et. al* [14] in their topology discovery tool Rocketfuel, use AS router clustering and BGP routing tables to direct probes in a way which exploits the assumed rule that a packet coming to an AS from network N1 with a next-hop-network being N2 will always take the same path through the AS.

These algorithms can reduce the amount of needed traceroutes by orders of magnitude [14] making the measurement unobtrusive and able to execute in one day. Of course, these techniques makes the sampling a non-trivial task in the topology discovery effort.

## 1.4 IPID

**Description.** IPID is a 16-bit field in IP packet header, originally referred to as Identification in RFC 791 [24]. It was introduced due to fragmentation of packets. If packet has to be fragmented, each fragment of the same packet has the same IPID value. This helps in reassembling the packets at the destination host.

**Why IPID is interesting in general.** The design of TCP/IP stack lacks support for measurement of many crucial network characteristics, yet even the available support is being less and less provided due to security reasons (network characteristics, topology being one of them, are considered confidential and some properties of TCP/IP protocols are being exploited for malicious attacks). This even led to design of special measurement protocols, like IPMP [25] or Hash-based IP traceback [26]. However until such protocols are widely supported, community has to operate with what is still available.

As various other IP packet header fields and options came under the spotlight of the research community with intention to use them for measurement of network properties, recently many experiments show the usefulness of IPID field.

**Uses in measurement.** It was used for:

- Inferring the amount of internal (local) traffic generated by a server, the number of servers in a large-scale, load-balanced server complex and the difference between one-way delays of two machines to a target computer [27].
- Counting of NATted hosts [28].
- Idle port scanning [29].
- Passive OS fingerprinting [30] and active OS fingerprinting [31].
- Alias resolution [14], [9], [5].

**Implementation.** There is nothing stated in RFC 791 about how the IPID distribution mechanism should be implemented. Therefore its implementation depends on decisions made in various operating systems. It may be simple incremental counter, either incrementing by 1 or by some constant. It may

also be a pseudo-random number, or it may even be a constant (at least initially set by OS). Usually there is a global counter, sometimes there is a separate counter for each interface or network stack running [32], [28].

**Observed behavior.** Usually, to exploit the IPID field, the mentioned techniques needs the IPID distribution mechanism to be implemented as a sequential counter. Therefore a throughout analysis have been done in various research works about how IPID is behaving in Internet [8] [5] [27].

Before looking at measuring results it is important to emphasize that as IPID is a 16-bit field, there are only 65536 possible values. Any measurement that tries to determine the rate at which an IPID counter increments has to deal with the fact that if measurement takes too long or the host increments counter too rapidly, the counter will eventually wrap. If counter was reseted during measurement, it will probably be incorrectly presented as pseudo-random counter. Analogically, if a counter is pseudo-random and successive probes coincidentally solicit responses with increasing IPID values, such a counter will incorrectly be presented as normal, or high-rate counter.

**Distribution of counter implementations.** While developing RadarGun [5] alias resolution tool, Bender *et al.* observed following distribution of IPID implementation mechanisms among 9 056 hosts.

| | |
|---|---|
| Unresponsive (less than 25% replies) | 4 240 (46,8%) |
| Linear | 2 841 (31,4%) |
| Non-linear | 968 (10,7%) |
| ICMP Destination Unreachable | 698 (7,7%) |
| IPID always 0 | 208 (2,3%) |
| Reflects the IPID of probe | 101 (1,1%) |

**Figure 3:** Results for hostname heuristics.

**Distribution of counter rates.** The rate at which counter is incrementing depends on implemented mechanism (e.g. some Windows family operating systems increments by 256) and on actual host's load (the more packet it sends the more times it increments the counter).

In measurements executed by Bender *et al.* [5], no host incrementing the counter linearly was incrementing at higher rate than cca 900 per second.

In results of measurements by Burch [8], 1% of 102 225 hosts were constantly sending IPID of value 0. After discarding these hosts, 79% of hosts were incrementing at a rate slower than 10 per second and 96% change at slower than 100 per second.

## 1.5 Loose source routing and record route option

**Description.** An IP packet header [24] may contain a field named Options. Among others, there are following options:

- Loose Source and Record Route
- Strict Source and Record Route
- Record Route

**Loose Source and Record Route.** If this option is present, it is followed by a list of via-points. These are IP addresses of routers the packet have to visit. Initially, the packet is routed in a standard way, based on the Destination Address field of the header.

If the destination is reached, the first address from the via-points list is set as new Destination Address value. Current router address will be placed at the beginning of the via-points list, instead of the first via-point taken. This replacement ensures the packets has the same size as before.

Finally, the pointer that points to the next via-point is incremented to point to the second address in the list. The packet is then routed as before, based on the Destination Address field of the header, until it arrives to that destination, and the process repeats.

After the packet visits the last via-point in the list, the pointer will point to address outside the option field. Such packets are routed by the Destination Address field, meaning the last via-point actually have to be the desired destination.

**Strict Source and Record Route.** This option is different from the previous in only one rule: the specified via-points represents the exact path the packet have to take, meaning the router always needs to have the next via-point as direct neighbor.

Because of its strictness, using this option is only possible if exact knowledge of the network topology is available. Moreover, the limit of the overall IP header size also limits the number of possible via-point to 9, thus this option can only be used for navigating the packet for 9 hops. Therefore this option is rarely used.

**Record Route.** This option enables the recording feature of the previous presented options for standard packets (without loose or strict routing). Each router the packet traverses inserts its own address into the list. The size of the list is initially set by the source of the packet and is initially empty. If some of the routers finds out that the list is full, it just forwards the packet as normally.

A question may arise, which one of router's addresses is inserted into the list. The option definition is stating that:

> The recorded route address is the internet module's own internet address as known in the environment into which this datagram is being forwarded. [24]

This basically means that router should insert the address of the outgoing interface.

# 2 Related work

**Summary.** In this chapter several known IP alias resolution techniques will be presented. These techniques may be divided into two categories: fingerprint methods and inference methods (division based on [32] is used although other exists). Finally, some auxiliary techniques will be discussed, such as how to split alias candidates before probing, or how to infer subnets to help alias resolution.

## 2.1 Fingerprint methods

**Description.** Fingerprint methods are based on active probing and subsequent analysis of collected packets. The aim is to provide evidence, that several packets were generated by the same host (e.g. incoming reply packets with different source address) or by several hosts with common properties (e.g. by hosts with the same initial TTL).

A general disadvantage of probing-only based methods is the fact that these depends on host's responsiveness to probes. Without responding routers (to the particular method), these methods are completely ineffective. Also, network changes during measurement may introduce errors.

### 2.1.1 Source Address method

**Origin.** Sometimes also referred as UDP technique, this method was used by Pansiot and Grad [9] as a first alias resolution technique ever.

It is based on an implementation characteristic of ICMP Destination Unreachable messages, that was described in detail in previous chapter.

**Method description.** An UDP probe is sent to some high numbered port where no service is assumed to be listening. Whatever public interface of a router is queried by UDP probe, router always responds (even if the interface queried is not the same as the one by which the query came). It responds with ICMP Destination Port Unreachable, and the reply is sent back by the same interface the query came from. While creating the reply packet the router

inserts the IP address of this outgoing interface to the the Source Address field of IP header.

This behavior can be exploited in a following way:

- Consider router having interfaces A and B.
- We query the router for interface B.
- Because of the physical location of the measurement host, the query arrives to router via interface A.
- Router replies, inserting A as the Source Address.
- After obtaining reply we see, that while sending query to interface B, interface A replied, thus we have the evidence that A and B are aliases.

**Advantages.** Main advantages of this method are simplicity and the fact that only one probe has to be sent to each obtained IP Address. Another strong advantage is that this method is not susceptible to false positives or false negatives.

**Disadvantages.** Not all routers respond to UDP packets in general (responsiveness to UDP probes was discussed in previous chapter). Moreover, not all routers respond with valid Source Address values. This may lead to inability to find aliases for such router, or to false positives, therefore such measurements have to be discarded. Examples of observed invalid values are:

- IP addresses from private address space [33].
- Invalid IP addresses, i.e. 0.0.0.0, 0.2.0.0, etc.
- IP address being always equal to the original probe destination.
- IP addresses from “dark” address space [34].
- IP address of the vantage point that sent the probe.

**Improvements.** Tangmunarunkit *et. al* [16] used this method in the Mercator tool. They introduced two modifications:

- Probes were sent multiple times (over long time period) to discover eventual backup paths and route changes.
- Loose source routing [24] was used to simulate multiple (geographically disparate) vantage points. This contributed to the method because

probes tend to come from various “sides” (various interfaces) of the router, thus more aliases were collected.

### 2.1.2 IPID method

**Origin.** Spring *et. al* [14] first introduced a method that exploits IPID mechanism for alias resolution. Because this method proved to be successful, it was throughly studied [9] [35] [32] and often new methods were compared to it [35] [37].

**Method description.** As described in previous chapter, IPID mechanism is often implemented as global incremental counter. Thus if two successive probes are sent to router, IPID values in replies to this probes will also be successive.

If the difference between IPID values is more than 1, it is usually because router was also replying to other packets[6] between replying to first and second probe. Therefore there is some value representing the maximal gap that may be observed between IPID values of successive probes to consider the replies to have common source router. It is however not useful to sent probes immediately one after another as ICMP replies sent by routers are often rate limited.

The resulting algorithm is a heuristic, actual parameters used in Ally (tool developed by Spring *et. al* that uses this alias resolution method) are as follows:

- First two probes (yielding x and y IPID values) are sent. If $|x-y| > 200$, interfaces are not considered aliases.
- If $|x-y| < 200$, a third probe is sent to prove whether IPIDs are generated in-order. If so, interfaces are considered aliases.
- Whole process is repeated again at later time, to minimize false positives when two measured routers coincidentally sends in-order IPID values.
- Rate-limiting of ICMP messages is dealt with this way:

[6]Directed to router on network Layer 3, not the packets belonging to the network traffic the router is processing. For such “routed” packets, the IPID is not changed.

> If only the first probe packet solicits a response, the probe destinations are reordered and two probes are sent again after five seconds. If, again, only the first probe packet solicits a response, this time to the packet for the other address, the rate-limiting heuristic detects a match. When two addresses appear to be rate-limited aliases, the IP identifier technique also detects a match when the identifiers differ by less than 1000. [14]

This rate-limiting heuristics alone (without the IPID value check) is sometimes presented as a standalone alias resolution method. Due to the fact that UDP and ICMP are unreliable protocols, missing packets (probe replies) should not be considered a proof of aliased interfaces, and no tool actually uses this method alone.

**Advantages.** IPID method has false positives, but are successfully minimized by second run. It was popular because it resolves more aliases than UDP Source Address method (linear incremental IPID counters seems to be more common than altering source address) [5].

**Disadvantages.** False positives count is small. False negatives are also possible, for instance if router uses random counter implementation or it increments IPID counter too fast or it has separate counters for each interface. Main disadvantage, most criticized, is the prohibitively high number of needed probes, $O(n^2)$, where n is number of IP addresses to test. This is partially solved by initial splitting of entry set of IP addresses, nevertheless, on an Internet scale this is still too much generated traffic. Another disadvantage is the probabilistic nature of measurement.

**Improvements.** Feamster *et al.* [36] used this method during their research of reactive routing. Used heuristics was simple: they repeated the IPID test 100 times. If the test was positive for more than 80 times, the tested interfaces were considered aliases. It seems to be a bit ineffective, however they also used a simplified method for choosing alias candidates, therefore the effort may have been appropriate to achieve the desired level of confidence.

Botta et al. [37] proposed some modifications of the Ally tool (they include a packet retransmission mechanism) for better multi-thread support as it is not trivial to run IPID method in parallel due to rate limiting. Implemented modifications are not described in detail in the paper.

Modifications used by Jimenez *et al.* [38] were based on the presented simulation of probability of false positives with a number of packets sent. Improvements like increasing the number of packets sent and using a static time offset between probes actually approaches the Velocity Modeling method (described later).

In reaction to disadvantages of the Ally tool, Bender *et al.* [5] recently introduced new technique based on IPID mechanism. The new method is called Velocity Modeling, and was implemented in RadarGun tool. It will be presented as a standalone method.

### 2.1.3 IPID Velocity Modeling

**Origin.** The original IPID method was developed for measuring ISP-sized networks. Especially because of number of probes increasing with the square of the number of discovered interfaces, with the ambition to discover the topology of Internet, IPID method becomes prohibitively ineffective unless supported by some sophisticated splitting algorithm[7]. This led to rethinking[8] of the IPID method in the paper “Fixing Ally's Growing Pains with Velocity Modeling” by Bender *et al.* [5] and development of the RadarGun alias resolution tool.

**Method description.** This method is based on a comparison of the samples from the IPID counters of two router interfaces (alias candidates).

The IPID value of each candidate IP address is probed several times. Probes does not have to be sent in a strictly regular manner and there is no time limit for the overall measurement.

Authors of the paper assume, that for a set of 500 000 interfaces, overall measurement should take less than 20 minutes with a single vantage point with 10Mb/s connection (although its in question whether such aggressive measurement is acceptable).

All interfaces should be probed in parallel, because comparing IPID values of two interfaces is more accurate if measured values overlap in time.

[7] Splitting algorithms are described later in this thesis.

[8] Niel Spring, one of authors of the original IPID method (Ally tool) was a co-author on this paper.

While modeling the IPID change function, it is vital to be aware of wrapping counters. During the first few probes, RadarGun estimates when the counter will reset (when in time). After this initialization, if a probe reply bears smaller value than the previous probe reply, RadarGun assumes a counter reset. If no reply have arrived until the end of the estimated reset interval, again, RadarGun assumes counter reset. Each reset adds 65536 to the value of modeled counter of the interface, therefore the counters are modeled as monotonously increasing.

After the measurement phase ends, the actual alias resolution comes into place. When comparing two interfaces, their sets ($S_A$ and $S_B$) of IPID values sampled in time should overlap on a time scale. Therefore the overall set of samples for this two interfaces can be divided to:

- Head – samples of $S_A$ collected before any samples of $S_B$ (or vice versa).
- Tail – samples of $S_B$ collected after any samples of $S_A$ (or vice versa).
- Middle – samples between head and tail (where samples from $S_A$ and $S_B$ overlap in time).

To express a single value indicating the relation of one IPID counter model to another, a property called distance (of modeled IPID velocities) is introduced.

The distance is computed in a following way:

> For each sample (t, id) in $S_A$ U $S_B$, we compute the distance between id and the expected value of the other IP ID at time t interpolated from the corresponding set of samples. The distances are summed across all samples, and divided by the number of samples to yield an average distance per sample. First, RadarGun sets a variable sum to 0. To calculate the distance of a sample ($t_H$, $id_H$) in the head, RadarGun estimates the value of B's IP ID at time $t_H$ using the linear approximation of $S_B$ to get an estimate $id'_H$, and adds $|id'_H - id_H|$ to sum. RadarGun executes a similar process to compute the distances between samples in the tail.
>
> For samples in the middle, RadarGun is able to make a more accurate estimation. Let ($t_{A,1}$, $id_{A,1}$) and ($t_{A,2}$, $id_{A,2}$) be samples in $S_A$ and ($t_B$, $id_B$) be a point in $S_B$ such that $t_{A,1} \leq t_B < t_{A,2}$. The estimated value of $id_A$ at time $t_B$ is interpolated based on the two points in $S_A$:
>
> $$id_A^{est} = (id_{A,2} - id_{A,1}) \frac{(t_B - t_{A,1})}{(t_{A,2} - t_{A,1})} + id_{A,1}$$
>
> $$sum += \left| id_B - id_A^{est} \right|$$
>
> Let

$$\Delta_{A,B} = \frac{sum}{|S_A \cup S_B|}$$

be the average distance between observed and expected IP ID per probe. If two IP addresses have a small $\Delta_{A,B}$ they are likely to be aliases, whereas a large $\Delta_{A,B}$ indicates that the addresses are not aliases. [5]

The actual settings of probing was following:

- 30 probes were sent to each interface.
- Interval between samples of a single counter was 34 seconds (actually an artificial limit of architecture).
- Maximal IPID velocity distance of interfaces considered aliases is 500.
- Minimal IPID velocity distance of interfaces considered non-aliases is 2000.
- Every comparison that yields values of IPID velocity distance between 500 and 2000 is considered “undetermined”.
- If less than 25% of probes returned, the interface is marked unresponsive.

**Advantages.** The main advantage is a very low count of probes (O(n)) needed when compared to original IPID method ($O(n^2)$).

As the probes does not need to be sent in a short time interval, the method is far less vulnerable to rate limiting and packet loss (there is an upper limit for this interval however).

For IPID counters implemented as pseudo-random, this method does not conclude anything. This is an advantage compared to original IPID method, as Ally concludes with high probability that such interfaces are non-aliases.

The method is not dependent on number of vantage points or on previously obtained traceroutes. It only needs a set of IP addresses.

**Disadvantages.** This method is ineffective for resolving pairs of interfaces where one of the compared interfaces (or both) does not have IPID counter that can be modeled as linear. This may be more than 10% of all interfaces (as shown in previous chapter).

It is also prone to errors in case of sudden change of IPID counter rate (such changes have been observed by the authors).

Also if probe replies does not arrive continuously (e.g. because of delays, although authors assume that the chance is minimal if the interval between probes is large enough), it triggers artificial counter wraps, thus is leads to errors.

As the two IPID counters have to be computationally modeled in the comparison process, the method involves some processing time, which is higher than in other fingerprinting methods.

Keys in [35] arguments, that although Velocity Modeling method does not have such a scaling difficulties as IPID method, still these difficulties may be prohibitive. As more routers have to be sampled, the interval between probes to the same router will have to increase. This however increases the probability of a counter wrap or even multiple wraps. Thus number of wraps inferred may be overestimated or underestimated, causing erroneous results in the analytical phase. This may be avoided with multiple vantage points, however, increasing number of vantage points is an scalability issue (not mentioning such vantage point will need synchronized clocks).

**Improvements.** In a recent presentation, Keys [39] suggest using TTL-limited probes instead of direct probing to improve response rate.

### 2.1.4 Record Route method

**Origin.** Sherwood and Spring presented this method while developing the Passenger topology discovery tool and the Sidecar platform it relies on [40]. The same method was described by Botta *et al.* [37] when developing the PingRR alias resolution module in their Hynetd topology discovery tool.

**Method description.** Sidecar is an engine for injecting probes to standard TCP streams. It deals with many challenges (connection tracking, probe identification, RTT estimation, rate limiting etc.) which will not be described in detail.

Passenger implements the logic of issuing probes. It uses traceroute like approach while enabling the Record Route option of IP protocol. PingRR uses similar approach.

This leads to two addresses obtained from each router (if router responds to these techniques):

- IP address which was in ICMP Time Exceeded message (incoming interface).
- IP address which was inserted because of Record Route option (outgoing interface).

Because incoming and outgoing interfaces of a router will have different IP addresses, obtaining both means successful alias resolution.

**Advantages.** In tandem with Sidecar, Passenger has a valuable advantage of issuing probes in a way, that it could be not distinguished from the normal traffic. This means that the measurement can avoid IDS alarms triggering, “suspicious traffic” logs and following abuse reports.

Furthermore, if there is enough confidence in the correctness of IP address alignments (see disadvantages), the alias resolution is as accurate as Source Address method.

Passenger has other advantages (e.g. discovering routers hidden by MPLS or the ability to resolve load balancing in traceroutes). These are very interesting, however not closely related to alias resolution.

**Disadvantages.** The main disadvantage of this method is its complexity and the uncertainty of the result. The problem that Sherwood and Spring observed is various implementations of the Record Route mechanism among routers. The same problem was observed by Botta *et al.* [37].

Some routers insert their IP addresses to the Record Route list only if the packet is to be transmitted forward. Some others also if the packet is to be discarded because of TTL being 0. Moreover, some routers does not decrement TTL (or only under some conditions) while some does, but not always update Record Route list.

Consider, that one traceroute may contain various routers with various implementations. The result is, that the alignment of the address in ICMP Time

Exceeded message with the address in Record Route list is a challenging task and because the rules by which various routers acts are unclear, this task has uncertain result.

Sherwood *et al.* [41] reports that because of this complicated aligning of traceroute data with Record Route data, 40% of data sampled by Passenger were unusable, moreover, 11% of aliases inferred from the rest where false positives.

Another disadvantage is that some routers discards packets with IP options. In recent paper on Velocity Modeling technique [5], Bender *et al.* stated that Record Route method discovered 11% of tested aliases, the IPID method contributed the bulk. This indicates that this method can hardly be used as a standalone alias resolution method.

**Improvements.** Sherwood *et al.* [41] recently released a paper describing a topology discovery tool DisCarte, which is based on aligning of traceroute data with Record Route data and validation against several rules. The tool uses disjunctive logic programming (DLP), a logical inference and constraint solving technique. This method promises better accuracy and completeness, however it is (as for now) too complex and expensive in terms of CPU time. Authors states that the solution for a topology measurement data containing 379 sources and 376 408 destinations will cost 11 CPU years on a 341 processor Condor cluster. Therefore this method will not be inspected in detail on this thesis.

## 2.2 Inference methods

**Description.** Methods which does not use probing are considered inference methods in this thesis. If there is enough confidence in the accuracy of collected traceroutes, some methods can infer aliases just from this dataset.

The crucial advantage of these methods is the fact, that no probing is needed, thus these methods do not introduce any additional overhead to the network (after traceroute collection is finished) and can discover aliases even for unresponsive routers. This is particularly important because the relative amount of unresponsive routers is expected to grow as the Internet service

providers limits the possibilities of measuring internal structure of their networks.

A common disadvantage is based on the fact, that these methods (without additional probing used) rely on their assumptions about network design practices. These assumptions are usually true only with some probability.

### 2.2.1 Graph method

**Origin.** Spring *et. al* introduced this method in a paper called “How to Resolve IP Aliases” [32].

**Method description.** First, a directed graph have to be built from the collected traceroutes. Afterwards this graph is analyzed by the Graph method. The method is based on two assumptions:

- There are only point-to-point links between routers.
- There are no loops in traceroutes.

If assumptions are considered valid, two rules can be defined for alias resolution in the graph built:

- Common Successor Rule: If an interface has more predecessors, these are aliases (because of point-to-point links).
- Same Traceroute Rule: Interfaces present in an traceroute can not be aliases.

**Advantages.** This method is simple and the Same Traceroute Rule can be used to narrow the probing with the fingerprint approaches.

**Disadvantages.** None of the assumptions of this method is actually valid. Real network topology contains subnets and MPLS networks and traceroutes may contain loops or invalid links. With increasing use of MPLS, this yields increasing error in the inference process.

Probably because of these misleading assumptions, this method is observed (by the authors) to have high false positives rate. It is therefore not recommended as a standalone method. However, in case of unresponsive routers it provides a possible solution and it may be used as a method for generating alias candidates for probing.

The Common Successor Rule (which actually discovers the aliases) exploits situations where two traceroutes overlap in some nodes. It is not clear how often this happens as two traceroutes, even over the same path but in opposite direction, may not contain any equal interfaces. This compromise the completeness of the method.

**Improvements.** If the techniques of subnet inferring and MPLS networks detection will be incorporated to this method, the Common Successor Rule may become less prone to erroneous results.

### 2.2.2 AAR, APAR and kapar

**Origin.** Gunes and Sarac proposed the AAR (Analytical Alias Resolver) method first in the paper “Analytical IP Alias Resolution” [42]. It this paper, the method is based on aligning two-way traceroutes based on searching for consecutive IP Addresses. It only assumes point-to-point links. Later [43] the same authors improves this method proposing APAR (Analytical and Probe-based Alias Resolver), this time using subnet inferring (later described by authors in detail in [44]) to align overlapping parts of traceroutes. Moreover, they add a Ping probe to measure the TTL distance of each interface.

**Method description.** First, the set of IP addresses in collected path traces is analyzed to identify candidate subnets (this technique is described in detail later in this thesis). The IP alias resolution process of APAR defines two phases:

1. All identified subnets are used to infer IP aliases, starting from subnets with best coverage in traceroutes (there is more confidence in these subnets). All rules of the algorithm are applied (described later).
2. Only /30 and /31 subnets (point-to-point) are used, and the Common Neighbor rule is not applied.

The idea behind the algorithm is, that if two symmetric traceroutes are properly aligned, we can see two interfaces of each router.

Consider having two routers with two interfaces: router R1 with interface $R1L_{eft}$ and interface $R1R_{ight}$ and router R2 with interface $R2L_{eft}$ and interface $R2R_{ight}$. Consider these routers separate two networks, network A, being on the left side of the router R1 and network B, being on the right side of router R2.

If a traceroute is sent from a host in network A to a host in network B, the traceroute output will contain the $L_{eft}$ interface of both routers R1 and R2 (because the intermediate probe of the traceroute this routers from $L_{eft}$ interface with TTL being 0, thus router replies with the ICMP Time Exceeded message back via this interface). Analogically, if the traceroute from network B to network A will be issued, its output will contain both $R_{ight}$ interfaces.

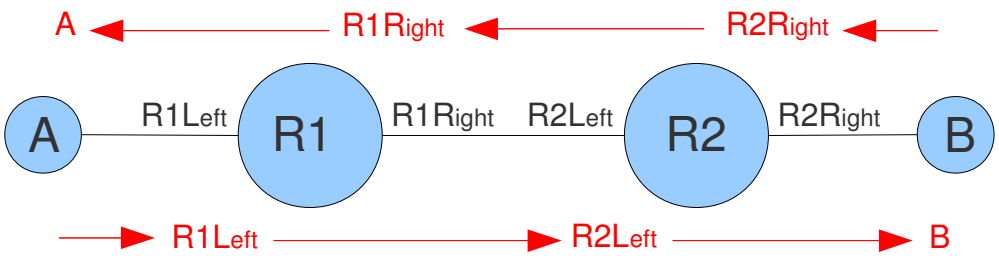



**Figure 4:** Non-aligned traceroutes

A question now arises, how to automatically infer, that interfaces $R1L_{eft}$ and $R1R_{ight}$ are aliases. Here, the IP address assignment practices comes handy. Because of the fact that public IP addresses are not to be wasted and because of guidelines for IP address allocations (RFC 2050 [45]) the network constituted from interfaces $R1R_{ight}$ and $R2L_{eft}$ will probably be a subnet with network mask /30 (or /31 as defined in RFC 3021 [46]). Therefore, IP addresses of these interfaces may be, for instance, 192.5.89.10 and 192.5.89.9.

During real analysis, we don't know that these two traceroutes (A to B and B to A) are symmetric. However, after using subnet inferring technique, we can infer that IP addresses 192.5.89.10 and 192.5.89.9 are in one subnet, thus these two interfaces can be aligned and this part of the two traceroutes becomes symmetric. For example, traceroute A → $R1L_{eft}$ → $R2L_{eft}$ → B and traceroute A ← $R1R_{ight}$ ← $R2R_{ight}$ ← B becomes aligned to A ↔ $R1L_{eft}$ – $R1R_{ight}$ ↔ $R2L_{eft}$ – $R2R_{ight}$ ↔ B. From such aligned traceroutes we easily infer aliases $R1L_{eft}$ – $R1R_{ight}$ and $R2L_{eft}$ – $R2R_{ight}$.

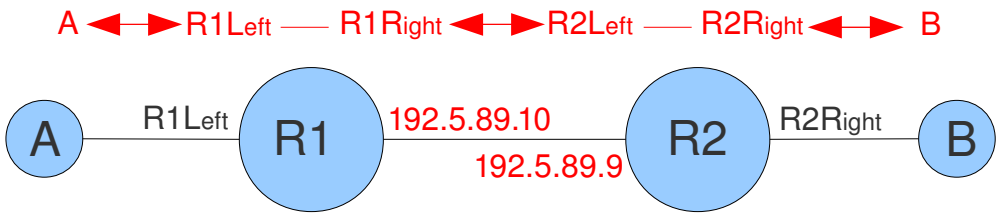



**Figure 5:** Aligned traceroutes

Authors presents several rules that avoid false positive in the inferring process:

- **No Loop** (the same argument as Same Traceroute Rule from Graph method) – If two interfaces are to be proclaimed aliases by the inference process, first, all traceroutes have to be checked, whether there is no traceroute where both these interfaces are listed. If such traceroute exists, aliasing is considered to be inaccurate.
- **Common Neighbor**

  > Given two IP addresses **s** and **t** that are candidate aliases belonging to a router **R**, we require that one of the following rules hold for setting them as alias:
  >
  > 1) **s** and **t** have a common neighbor in some path trace
  >
  > or
  >
  > 2) there exists a previously inferred alias pair (**b**,**o**) such that **b** is a successor (or predecessor) of **s** and **o** is a predecessor (or successor) of **t**
  >
  > or
  >
  > 3) the involved path traces are aligned such that they form two subnets, one at each side of the router **R**.
  >
  > [43]

- **Distance** – Two aliases should be at similar distances from same vantage point. The distance is measured by TTL from reply packets. The actual used threshold is not provided, but thresholds for this metric are discussed later in this thesis, where splitting techniques using this metric are studied.

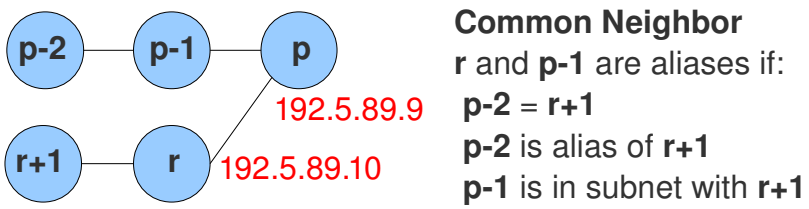



**Figure 6:** Common neighbor rule

The Common Neighbor rule is not needed in the second step of APAR, because in point-to-point link, if **p** and **r** are in the same subnet, **r** and **p-1** must be aliases.

**Advantages.** In presented results, this method discovers high number of aliases not discovered by probing based methods, while keeping low rate of false positives and negatives.

**Disadvantages.** The method depends on presence of traceroutes that have symmetric parts. Moreover, it only infer aliases for routers that are intermediate in the traceroute (not for routers being the destination of a traceroute).

If path asymmetry is present in the traceroute collection (a commonly observed property), the method can not find aliases along the path (as such traceroute does not have overlapping segments).

**Improvements.** Keys [35] improved the APAR algorithm in the “kapar” implementation in several ways. First, in some technical details of implementation which causes the APAR to be implemented as a single-pass algorithm and enables use of various data sources (for instance Source Address method or collected TTL's from multiple vantage points) to improve the accuracy of algorithm. kapar also improves the subnet inferring phase.

## 2.3 Hybrid approaches

**Description.** More researchers advocated the idea of using various known methods in concert to yield better accuracy, efficiency and completeness of

alias resolution. Moreover, some methods may be ineffective not from principle, but because of actual configuration of network, and the success of this methods then vary from ISP to ISP, thus are not suitable for Internet measurements unless accompanied by some other method.

For instance Gunes and Sarac in [43] show, while evaluating their APAR method, that APAR and IPID based method have each a tendency to discover aliases the other method can not, thus the way to accurate alias resolution is in combining these methods together.

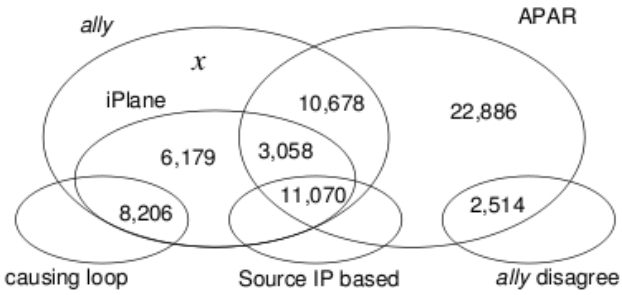



**Figure 7:** Comparison of used methods from Gunes and Sarac.

Spring *et al.* [32] also state, that combination of Source Address based, IPID based and Graph method finds more aliases than any of the techniques alone[9]. Moreover, some combinations may be more appropriate for some scenarios (e.g. not all methods, the DNS method for instance, are suitable for Internet scale measurements).

Keys [35] used hybrid approach with success [39] by using kapar (APAR based) and iffinder (Source Address based) tools together:

> Even on parts of the Internet where iffinder does not find any aliases, results for iffinder+kapar are better than for kapar alone. [39]

Keys also plans to add IPID Velocity Modeling to the solution.

In their Alias Filter algorithm, Hong-Hua *et al.* [47] combine various inference rules (mainly from Graph method), assumptions and filter rules[10] with Alias

[9] Only comparative methods were presented, as the real topology was not available.

[10] For instance that no two IP addresses from different ISPs or geographical locations (except boundaries) may be aliases.

Relation Validation phase based on IPID method. They also use DNS and geographical location datasets.

However, the motivation of some of the assumptions (e.g. that there are no two parallel paths between two routers in one directions) is unclear and although authors stated that the AF+ARV method missed less alias relations than other approaches, no proof or comparative study is provided.

## 2.4 Auxiliary techniques

### 2.4.1 Splitting the set of candidates

**Splitting.** It is always desired to keep the number of probes low. Here the splitting of the set of alias candidates can significantly help. In principle, all IP addresses are alias candidates to each other. The aim of the splitting techniques is to divide this initial set to subsets containing only IP addresses which are likely to be aliases.

**TTL.** When sending traceroute probe, an estimation of the TTL distance of the probed machine is obtained (number of hops on the reverse path from probed machine to probing machine). Also when obtaining packets from routers, it is possible to infer the initial TTL value.

The TTL heuristics suggests that:

- Aliased interfaces should have the same initial TTL value.
- Aliased interfaces should be in the similar TTL distance from each vantage point.

While the first suggestion does not disprove many candidates (there are only cca 6 or 7 common values of initial TTL), the second is more successful especially when in combination with some other method, a graph method for instance. However it is not easy to establish the TTL distance threshold. This distance is usually set to 1, this means that all IP addresses which are more than one “hop” from each other are not considered aliases. Spring *et al.* in [14] obtained following statistics considering TTL:

> Of the 16,000 aliases we found, 94% matched the return TTL, while only 80% matched the outgoing TTL (the TTL that remained in the

> probe packet as it reached the router, which is included in the response.) [14]

In their Alias Filter algorithm, Hong-Hua *et al.* [47] use the hop distance threshold value of 3.

Because it is very unlikely that addresses that are more than few hops from each other are aliases (at least without routing loops in traceroutes), this technique is popular and often used, because it reliably narrows the initial set while introducing only insignificant error. Moreover, the TTL values are often saved to the traceroute output, so no additional probing is needed.

**DNS.** Spring *et al.* in [14] developed a method which infer alias candidates by inspecting DNS names of the interfaces. Two interfaces, named for instance cityB-routerC-ifaceX-ispA and cityB-routerC-ifaceY-ispA are likely to be aliases. Therefore the initial set of candidates can be split by this “lexicographical adjacency”.

A critical disadvantage of this method is that the rules used to parse DNS names needs to be defined and maintained by human operator. It is unlikely that someone will obtain and maintain a database of such rules for all ISP's in Internet.

**Graph.** A graph method proposed by Spring *et. al* [32] may be used to obtain alias candidates. This method (particularly the Common Successor Rule) is often used (for instance by Feamster *et al.* In [36]) as it does not need additional probing and its assumptions are sensible. However, it has a low coverage, as it can only leverage a situation where two traceroutes merge at some point.

### 2.4.2 Subnet inferring

**Subnet inferring.** In [44] Gunes and Sarac presents algorithm for inferring subnets in traceroute collection. They also emphasize that detecting the subnets is beneficial for alias resolution and improves the accuracy of the obtained topology. For example, if subnets are not taken into account, the resulting topology will contain (ideally) a mash subgraph for all nodes in a subnet, instead of a single shared link. In another paper [43] authors build the APAR resolution method on this technique.

Obtaining subnet of an IP address will be trivial if the ICMP Address Mask Request and Address Mask Reply (RFC 950 [47]) messages will be supported by the routers, but unfortunately this is not the case, these messages are usually ignored.

Inferring the subnets is based on the assumption about the IP address allocation process used in practice. ISPs usually obey rules defined in RFC 2050 [45]. To test the correctness of the inference process, probes may be added to the algorithm to confirm whether the inferred subnet is real.

The algorithm proposed by Sarac and Gunes is iterative:

1. By combining IP addresses from the dataset, where first N bits of these addresses match, they form candidate subnets.
2. From these initial candidate subnets they recursively form smaller subnets while using four rules to infer, whether the subnet created is real.

The four rules are:

- **Accuracy** – If a loop-free and correct traceroute is assumed, all IP addresses in the output of such traceroute should be from distinct subnets. If some of the routers does not comply with RFC 1812 [10], it is not a correct traceroute and may contain two addresses from the same subnet. However, in this case those two addresses may be at most one hop from each other. If two addresses, believed to be in one subnet, are present in a traceroute with more than one hop between them, the subnet is not real.
- **Distance** – All IP addresses from within a candidate subnet should be in similar TTL distance from particular vantage point. These distances should not differ by more than one hop.
- **Completeness** – To prevent inferring of large subnet containing small number of IP addresses (which is not probable and can hardly be verified), subnets that have less than one quarter of IP addresses present in the dataset are ignored.
- **MaxFit** – if a subnet is considered real by previous rules, smaller subnets created from this one will be ignored (as the probability that

they will also match the criteria is high). Only exception is that two IP addresses from a /30 or /31 subnet are considered to be in /31 subnet.

In a different network topology sampling study by Siamwalla *et al.* [48] authors propose two heuristics:

- **Subnet guessing using broadcast pings** – Which literately tests inferred subnets (from /31 to /7) for each IP address by sending broadcast pings. This method introduce a significant overhead and is not reliable, as many routers discards broadcast pings considering it a Smurf attack and some replies to such ping themselves (instead of hosts in the pinged subnet). Moreover, this method is slow.
- **Subnet guessing from a cluster of addresses** – If a cluster of IP addresses is present in a topology (IP addresses in one hop distance behind a common router interface) its subnet can be inferred by using bitwise AND and bitwise OR on these IP addresses. However, if all hosts lie in the higher end of the address subnets space, this method can not decide on the subnet mask.

As shown in [44] vast majority of the subnets are of size /30. Subnets larger than /28 are rare.

## 2.5 IP alias resolution tools

**Summary.** A quick overview of existing alias resolution tools. Authors and methods and important algorithms used by this tools were introduced in previous chapters, here only index-like listing is presented.

**Ally**

Uses mainly IPID method with some other metrics (Source Address method, Rate-limit technique, etc.). Developed for Rocketfuel project, later used in many others.

**iffinder**

Main method is the Source Address method, but also Record Route is leveraged and /30 subnet inferring heuristic is used.

**Mercator**

Uses Source Address method and improves the method by exploiting source routing.

**APAR**

The original proof-of-concept tool released by authors of APAR algorithm.

**kapar**

Optimized implementation of the APAR algorithm developed by CAIDA. Some additional heuristics are used: TTL from multiple vantage points and stricter subnet inferring supported with subnet broadcast probing.

# 3 The Jardinero tool

**Summary.** The practical output of this thesis, the Jardinero IP alias resolution tool, and its contribution to the field of IP alias resolution is thoroughly analyzed in this chapter. First, the design of the software is explained aside with the decisions that shaped it. Later the performance of its modules and real-world results are presented.

## 3.1 The objectives

The main objectives of the implementation of the new IP alias resolution tool were:

**Hybrid approach.** As can be clearly seen from the analysis of the available state of the art IP alias resolution methods, each one has significant disadvantages. Many authors [35] [32] [43] stated, that if the best possible accuracy and completeness is desired, at least one active (fingerprint) and one passive (inference) method has to be used. Active measurements are completely ineffective on unresponsive routers (which constitutes a too large fraction of overall router set to be ignored). Inference methods heavily depend on the quality of the input dataset (often hard to achieve, like symmetric traceroute paths) and are prone to high false positive rate if not revised with data from active measurements.

**Efficiency.** Many of proposed methods were alone too ineffective to be used on networks with thousands (and hundreds of thousands) of nodes. For inference methods, the optimization leads to effective computability, for active methods, some splitting algorithm have to be used to focus the measurements to sets of nodes small enough making the measurement possible.

## 3.2 Design

**Summary.** The explanation of some key decisions made before and during the development of the tool.

### 3.2.1 Used alias resolution methods

**Source Address method.** Virtually every serious alias resolution effort (e.g. [14] [9] [35]) uses this method. The reason is its peerless accuracy and the fact that it scales linearly for large networks. The disadvantage, prohibiting a sole use of this method as an active measurement part, is its limited completeness (which is also dependent on the number of vantage points).

Still, as stated, it is a very effective method and these properties together with the confidence in its results makes it an essential part of the project.

**IPID Velocity Modeling.** Although this method is not strong in completeness (it only resolves routers with linear counters) it has crucial advantages. Aside from good accuracy and robustness, its input is only a set of IP addresses, it is neither dependent on number of vantage points nor on any conducted traceroutes, what makes it a prominent (if not the only one) method to use in some situations.

**APAR.** The main disadvantage of APAR is its need for a high-quality input dataset. Yet its results are very good [43] [35] and in an unresponsive environment, it is virtually the only effective method. As such, it constitute the inference-based part of the hybrid approach of the project.

### 3.2.2 Not used alias resolution methods

**Active measurements.** The IPID method was not used as it was outclassed by IPID Velocity Modeling in virtually every aspect. The Record Route method seems to be too expensive on both, implementation and resources [41] [35], to be effectively used, and without proper implementation it will not provide credible results.

**Inference methods.** The Graph method was not used because its high false positive rate makes it only suitable for providing a set of candidate IP addresses, and the candidate set was created in a different way in the Jardinero project.

### 3.2.3 Platform

**Java.** The reason for using Java is the ease of deployment and integration with very good possibility of monitoring and debugging the application. The only non-Java dependency aside from PostgreSQL is the Jpcap library that provides the lacking support for low-level networking in Java. This decision makes the project easily manageable and also nearly instantly cross-platform.

**PostgreSQL.** The decision to use traditional SQL database was mainly the transparency requirement and the research nature of the application. Although the relational database is not really needed by most of the used algorithms (and it is often not even used in this way) it gives the researcher the ability to look at the data at nearly any phase during the alias resolution process and to issue the SQL queries as needed. PostgreSQL was chosen because it is open source and multi-platform (as such easily integrable to existing infrastructure), quickly innovated while stable and has a good support for network data types and functions.

### 3.2.4 Basic architecture

**Modular architecture.** The fact that the application is composed from more standalone tools or modules results naturally from the fundamental divergence among the needed operations. There is a need to import the initial or measured data to database (the Imp module), there is a need to analyze the obtained data and get the results (the Digester module) and there is a need for a component for active measurements (the Pulsar module). The database server along with prepared stored procedures / functions may also be counted for a module.

**Communication.** To further support the distributed nature of the application, the Pulsar module communicates with the rest of the modules (and between its own submodules) purely by means of compressed binary files. This eases its secure deployment far from other modules, while keeping the communication overhead low.

Two remaining modules, Imp and Digester communicates with the database module via JDBC (type 4) protocol. For efficiency, the communication is cached with batch and fetch operations.

**Database.** The database module act not only as a storage. Many of the application's logic is written in the PLpg/SQL language. This avoids storing large data in memory and copying them across network in case the logic will be purely written in Java. More important, during some procedures, the algorithms works with other data in database (for example, in case of resolving hops with anonymous routers, the import procedure analyzes all the already imported hops). Such interactions can not be done efficiently elsewhere, only in the procedures on the server side.

**Imp.** The module imports initial datasets (traceroute dataset, AS to AS relationships, etc.) and results of active measurements. It may run on a per file basis, or as a daemon watching a directory for incoming files. Usually it parses data from input files and calls import procedures, sometimes however, it also processes the data.

**Digester.** First, the Digester module serves as user interface to database so the user don't have to work with the database manually. Second, it implements algorithms that would be too complicated to compute in PLpg/SQL. Finally, it outputs processed results to input files for Pulsar, or to graphs and plots.

**Pulsar.** Provides all needed active measurements, fingerprinting and IPID sampling. It is divided into two submodules: sender (sending probes specified in input files) and captor (preprocessing input files, capturing probes, creating output files). These submodules may run as a single application or separately, even on different computers, if needed.

### 3.2.5 Functionality

**The main contribution.** Although implementing the most contemporary alias resolution methods (and improving them as possible or as necessary) was a non-trivial challenge, it only constitutes roughly a one third of the project (and its complexity). The aim of the project was not just to create an

implementation of each chosen method, with all the bells and whistles, but to provide a support for every aspect of alias resolution effort, from the initial import of traceroute data, to the final interface clustering.

The focus of the work therefore lays also in procedures that obtains, analyzes, prepares and evaluates the data even before and after the alias resolution itself.

**The typical run.** To support the idea of the project's integration to some large-scale network measurement effort, a scenario for its use have to be created.

Assuming a large-scale measurement, at the beginning of alias resolution is a massive collection of traceroute data. To simulate this situation, raw data from iPlane project [15] were used as the input.

First step is the import of the dataset to the database module of Jardinero project. During the import, data will be processed as much as possible to accelerate later operations.

Next, alias resolution method will be used in a carefully engineered order to provide maximum efficiency. For instance, because the inference method, APAR, may profit from nearly every available information, it will be the last on schedule.

The IPID Velocity Modeling method has its limitations about how many nodes can be tested in parallel. Because the method repeatedly sends measurement probes to large number of IP addresses, one natural way to save resources during the process is to only sample IP addresses that responds to such probes. Another improvement that comes in mind is to split the large initial set of IP addresses to smaller subsets, by applying constraints (taken from from already present information) which defines what IP addresses may not be aliases.

To find out which IP addresses are responsive and to enrich constraint information, a separate process, fingerprinting, was implemented. The fingerprinting scales linearly with the input, yet it maybe unnecessary to fingerprint everything in the imported collection (as the alias resolution itself may not be required on the whole collection). Thus, before the fingerprinting the user selects a subpart of imported data. Depending on the size of the

dataset, it may also be a necessary, to support the later splitting needed for active measurements.

The overall order of procedures is as follows:

1. Import and preprocessing of traceroute data
2. Selection of the alias resolution input
3. Fingerprinting (including Source Address method)
4. Splitting
5. IPID sampling and IPID Velocity Modeling
6. APAR
7. Normalization (creating of IP alias clusters from all available information)

The import is provided by the Imp module, while the fingerprinting and IPID sampling runs from the Pulsar module. The rest of the procedures are provided by Digester module in concert with the database.

## 3.3 Implementation

**Summary.** The principles of the implementation of the Jardinero tool are explained with respect to the order of procedures presented at the end of previous chapter. Some technical details and notes on its usage are left for the user guide of the tool.

### 3.3.1 Import and preprocessing

**Input data.** There are following types of input files for Imp module:

- Binary file from iPlane project containing raw traceroute data [15].
- Text file containing traceroutes (format is specified in user guide).
- Text file from CAIDA containing definition of AS-to-AS relationships [49].
- Text file from CAIDA containing observed of AS-to-AS links [50].
- Text file from DIMES project containing observed of AS-to-AS links [51].

- Text file from CAIDA containing mapping of network addresses to origin AS [52].
- Text file from iPlane project containing mapping of network addresses to origin AS [15].
- Text file from countries.nerd.dk rsync server containing mapping of network addresses to origin countries [53].
- Binary file containing results of fingerprinting process.
- Binary file containing results of IPID sampling.

**AS relationships.** The files containing AS relationships may seem redundant, but in fact each contains some information not present in the rest. The relation of the files to each other is described by the diagram on the following figure.

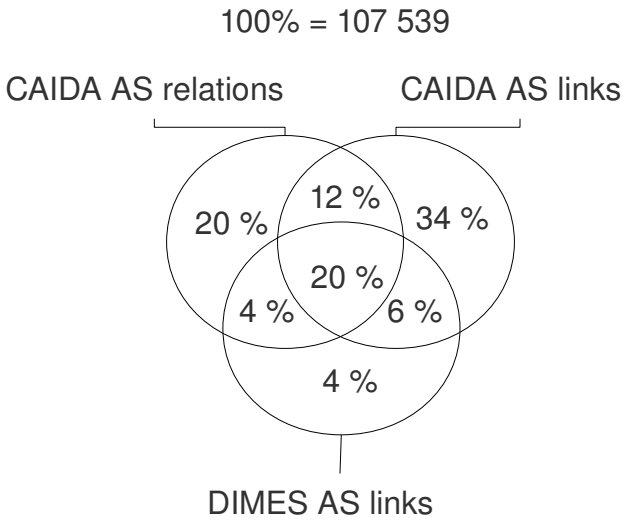



**Figure 8:** The AS relationships datasets

**Network to origin AS mapping.** On the following figure, similar diagram is presented for the CAIDA and iPlane files containing mapping of networks to origin AS.

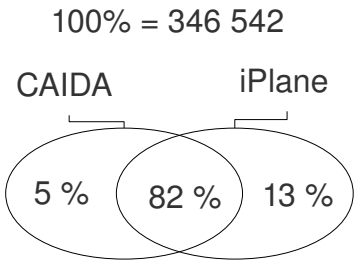



**Figure 9:** The network to origin AS mapping datasets

**Importing traceroutes.** Many procedures involved during the whole alias resolution process heavily depend on the traceroute dataset. Therefore the dataset have to be solid and that is far from what the raw traceroutes are. Also the amount of secondary information that have to be extracted from the traceroutes is significant. Together this makes the import of traceroutes a time-consuming process despite the improvement effort.

Following operations are involved:

- Every IP address is checked for validity. Traceroute with invalid address is imported, however, marked as containing invalid IP address. Invalid means:
  - It is from “zero” A-class network (e.g. 0.1.2.3) except 0.0.0.0 which is a mark of unresponsive router.
  - It is from private address space [33].
  - It is from multicast address space [54].
  - It is the address of the source or destination of the traceroute (naturally checked only for intermediate nodes).
- Traceroute is checked:
  - If it contains loop, it is discarded.
  - If it contains more than 2 successive unresponsive (anonymous) routers, it is trimmed and imported.

- Impossible subnets are extracted (as described in [44]). The inference rule is described in chapter discussing subnet inferring. Extracted information is later used in APAR.
- All hypothetical /24 subnets are extracted, again, this information is later used during subnet inferring phase of APAR.
- Unresponsive / anonymous routers are resolved with help of the bisimilarity rule:
  - As traceroutes containing more than two anonymous routers are discarded, we may observe following two scenarios:

    1. $IP_a$ – anonym – $IP_b$
    2. $IP_a$ – anonym – anonym – $IP_b$

    If, for a particular $IP_a$ and $IP_b$, such combination was never observed, IP address from the "zero" A-class network is assigned to the anonymous routers, and this combination is saved. For instance, a following record may be created: $IP_a$ – 0.0.0.1 – 0.0.0.2 – $IP_b$. Next time the combination of $IP_a$ – anonym – anonym – $IP_b$ is observed, the anonymous routers will automatically be assigned with 0.0.0.1 and 0.0.0.2 IP addresses.
- Every IP address is inserted to database along with the initial TTL of the reply packet and the result of the IP address validation. If the IP address is already in the database, these information are updated. The value of the initial TTL is guessed from the TTL of the reply packet included in the traceroute file for every address. The algorithm for the guess is following:

```
INPUT:
TTL /* The TTL in the reply packet */

OUTPUT:
InitTTL /* Approximation of the initial TTL value */

   1. if TTL < 96 return 64;
   2. if TTL < 192 return 128;
   3. return 255;
```

**Figure 10:** Initial TTL guessing algorithm

- The rationale behind this algorithm is taken from the popular Nmap tool [55]. Virtually only values 64, 128 and 255 are used (32 is very rare) and as the hop distance is rarely more than 20, the initial value can be easily guessed as presented.

- The distance of every IP address from the particular vantage point (that created the traceroute file) in the means of hops is saved. Also the TTL of the reply packet is saved here, as it is a value relative to the vantage point.
- Mapping of ID (assigned to the particular traceroute) to each of its IP addresses is saved (later used for the “no loop“ condition in APAR).
- Every unique 3-hop segment is saved (later used in APAR, it is an important improvement introduced by Keys [35] for efficiency reasons).
- The role of each IP address is inferred. The idea is, that traceroutes are typically issued for many destinations, but paths to those destinations does not differ that much. Therefore, IP addresses of destinations (usually hosts) will constitute a significant fraction of all discovered IP addresses, while what is usually needed is alias resolution of only routers. If only routers could be used for alias resolution, it may radically lower the number of IP addresses to scan. Moreover, scanning a host is much more likely to trigger IDS alarm.

  The algorithm for dividing IP addresses to hosts and routers is following:

  1. Source addresses of traceroutes are hosts.
  2. If the destination IP address is not already in database, it is considered host.
  3. Every other IP address (that is intermediate IP addresses in traceroute, or even the last one, if it is not the destination) is updated to be router.

  This way it is ensured, that if the IP address was “crossed” it will be labeled as router. If an IP address was not crossed, it will probably be host.

  If a router was a destination of some traceroute, it will be mistakenly labeled host, but only until it is crossed in some another traceroute. If it is never crossed, it will stay labeled host. However, it is unlikely to

happen that a router which was not crossed has an alias IP address in the dataset (and also it is unlikely that any alias will be discovered for such router even if it will be labeled as router).

**Rest of imports.** When importing fingerprints, if the fingerprint is a response to the UDP packet, the distance to the vantage point is revisited (if any traceroutes from this vantage point were imported) as it is assumed that latest imports contains most actual information. If no response to the fingerprinting packet was observed, this information is also saved as it is valuable. The Source Address aliases are not only saved after fingerprinting (the fingerprint is saved for both of the aliased IP addresses in this case) but also after the IPID sampling (if UDP packets were used and alias was found).

### 3.3.2 Fingerprinting

**Objective.** There are two main objectives of fingerprinting, to improve the possibility of splitting the input to active measurement and to reveal Source Address aliases. The more information that can be obtained for every IP address the higher is the possibility to split the data to subsets without increasing the overall number of probes that have to be sent and without missing any aliases.

**Extended fingerprinting.** Normally, only the hop distance, initial TTL or IPID counter slope were used as the base for splitting. However, all these information may not be enough for effective split. Here, it was assumed, that a Nmap-like fingerprint may provide additional information and that such data may prove to be very useful for this purpose, although it was in question, whether the routers in Internet are diverse enough in their TCP/IP stack implementation (not mentioning that it was assumed that enough routers will be responsive to probes).

Of course, a full-fledged Nmap scan [31] of a large network will not be possible. First, it is resource intensive, and second and more important, it is too obtrusive. Not only that such effort will probably have to deal with abuse complaints from network administrators; it is not acceptable from principle to disturb normal traffic with measurement (especially if such measurement is to be repeated regularly). It will probably not even be effective, as routers are

not regular hosts and it is not common for a router to have any ports open and typical network services running to be probed from regular vantage points.

Instead, only subpart of fingerprinting was used. Probes are sent to high numbered ports (where there is virtually zero probability that any service will be running). Whether this approach was successful will be analyzed in evaluation chapter.

**The implementation.** The fingerprinting method is implemented from scratch in the Pulsar module, but the fingerprints are roughly the same (in the means of used probe packets, tests of replies and output) as following fingerprints from Nmap: IE (ICMP Echo probe), U1 (UDP probe) and T5 – T7 (TCP probes to closed ports). The detailed description of probe packets is left for source code documentation (JavaDoc), the test that are executed on the obtained replies are described instead.

**Echo probe.** The reply to the echo probe is not valuable in the expected way. Actually, all fingerprints obtained were completely equal. It proved to be very useful though. Quite surprisingly, it was common that the initial TTL set to the Echo reply was not the same as for replies to TCP or UDP probes (these may also differ, but not as often).

Following analyzes are undertaken on the Echo reply packet:

- CD – The value of the Code field of the ICMP reply is inspected. It is expected to be zero, but may be another value in faulty TCP/IP stacks.
- DF – State of Don't Fragment bit.
- TTL – Value of the TTL field.
- INIT_TTL – A guess of the initial TTL value (algorithm for guessing is the same as was explained in description of import procedure).
- RID – IPID value of reply is checked whether it is the same as the IPID value of probe.

**UDP probe.** The UDP probe is crucial as it not only elicit reply which constitute fingerprint but also is the instrument of the Source Address method. This however, brings a bit of a complication into the process of matching

replies with probes. The matching procedure will be discussed but first, the measurement process have to be explained.

**The measurement procedure.** The Pulsar module has two parts, which will be called sender (sending packets) and captor (capturing packets). At the beginning of the measurement, the file containing the definition of the measurement is generated by Digester module and sent to the captor. This is a general procedure, more or less the same for all probe types. For the UDP probes however, there are few more steps while generating the file:

- Source and destination ports are randomly generated (from values over 49 152 to minimize the chance of probes to be evaluated as an attack).

  For UDP probes this port numbers act as a key. When an ICMP Port Unreachable packet is received, the captor searches the port numbers in its internal map to find out what was the original probe. Therefore the port numbers have to create a unique combination for each probe in a particular measurement.

  If only the destination port number will be randomly generated, it would not be possible to send more than roughly 16 thousand probes. If also the source port number is generated and used as part of the key, it gives more than 268 millions of combinations. The network card is used in promiscuity mode by Pulsar module, and so the application is not bound to any specific set of ports (although ICMP replies for UDP probes does not use port numbers anyway).
- A random 2-byte value is generated that serves as variable part of data payload.
- The generated data, port numbers and IP addresses are put together in a packet header and its checksum is computed (later used for RUCK fingerprinting test).

All these generated information along with the destination's IP address is saved under the key created from the source and destination port. The definition file is then sent to captor, which inspects the file, creates the internal structure needed for matching the captured packets with probes and inserts its own IP address to the file (to be used as source address by the sender). The updated definition file is transferred to sender, which sends all

specified probes in configured intervals. After all the probes were sent, special UDP packets – stop packets – are sent from sender to captor. These informs the captor that all probes were sent.

If, after the stop packets were received captor still miss some replies, it may create a new definition file with probes it saw no replies for, and transfer it to sender for re-probing. This may happen several times as needed. It is useful, as will be explained later, to be sure that the unresponsive addresses are configured to not respond and the lack of the reply is not due to packet loss, which is common in UDP communication.

**Matching replies with UDP probes.** Not only the source address of the reply may be unknown to the packet captor, even the source address of the probe, that should be included as part of the returned IP header of the probe, may not be correct (as observed by Burch [8]).

After an ICMP packet is received, following procedures are executed:

1. Type field is checked, only Destination Unreachable packets are interesting.
2. The byte length of the data field is checked. Data field should contain the IP header (20 bytes) of the original probe + 8 bytes of the original data payload [56].

   Often the payload is not included at all, while sometimes whole the payload is included (not only first 8 bytes).
3. If the included IP header is present, and the source IP address of the reply differs from the original destination, then an Source Address alias was found. If the payload is present, the ports may be used to match the packet with probe and then check whether the data in the payload are same as were sent, etc. This increase the confidence in the alias, because if the payload is not present and the proclaimed original IP address was not correct than such alias is false (Burch concludes this may be because of a faulty NAT).

**UDP probe fingerprint.** Whether an alias was found or not, the captor marks the IP address as disposed and creates the fingerprint. The fingerprint is tested for:

- TTL, INIT_TTL, RID and DF as in Echo fingerprint.

  The guessing of initial TTL is improved by using the TTL from the original IP header (the TTL set by sender as seen when the probe arrived to the destination). The value set by sender is, of course, known and by observing how many times it was decremented on the way to destination we may infer the hop distance to the destination. Adding this distance to the TTL in the IP header of the reply should give the initial TTL set by the destination.

  Still the value have to be “rounded” to the nearest possible value as the path asymmetry is very common in Internet and so the number of hops on the way to the destination often differ from the number of hops on the way back.

- IPL – Tests total length of the IP packet (may differ as the length of the returned original payload may differ).
- RIPCK – Tests if the checksum of the IP header is valid.
- RIPL – Tests the length of the original IP header.
- RUCK – Inspects whether the checksum of the returned original IP header is the same as was when sending.
- RUD – Inspects whether the data payload of the original probe was truncated or modified.
- UN – A check if the last 4 bytes of the ICMP Port Unreachable header are set to zero.

**TCP probes.** There are three types of probes to send. A SYN probe (normally used to initiate TCP connection), ACK probe (normally used to acknowledge ongoing connection) and FIN probe (normally used to finish the connection). For all three types of probes the RST packet is expected as a reply (used to drop connection), because no connection was established. In a way, it is not expected for FIN probe as it is not needed, however, many devices send the RST packet anyway. The RST packets used as a reply for a particular probe tends to differ, besides, there is also a difference among the TCP/IP stack implementations. The RST packets are tested for:

- TTL, INIT_TTL and DF, same way as with UDP and Echo probes.

- A – Inspects the acknowledgement number (whether it is the same as in probe or how it changed).
- S - Inspects the sequence number (analogously as the acknowledgement number).
- F – Creates string representing the TCP flags.
- Q – Warns of the presence of some rare modifications of the TCP header.
- RD – A checksum representing the data from the TCP packet, if any (some systems inserts error messages).
- W – Records the value of the TCP window size.
- O – String representing the options in the TCP header.

**Output.** The output of the fingerprinting is a string representing the fingerprint in the Nmap format (except that the initial TTL is not part of the fingerprint and is stored separately). Although it may be possibly used for inferring of the operating system running on the probed device, for the alias resolution process it is only important whether one fingerprint differs from the other.

### 3.3.3 Splitting and IPID sampling

**Motivation.** As explained, active measurements have to be targeted. For large-scale network measurements, a natural desire may be to only search for aliases within some autonomous systems (AS), within some countries or within some domain. Some of such constraints may not be easily used. The first two mentioned however, were used in this project with the help of the datasets described in the documentation of import procedures.

Unfortunately, a decision to target the measurement to a specific country or countries may not be enough limiting, consider country like U.S. or Germany. The same holds for AS constraint, consider a Tier-1 AS or the fact that if it is desired to search for all aliases of all devices of a particular AS, the networks of the peering autonomous systems should be included (and some autonomous systems, for instance GÉANT2, peer with other systems virtually on every router).

Hence, after the countries and/or AS constraint was applied, further splitting constraints (attributes) have to be found.

This is critical only for the IPID sampling. The fingerprinting has a linear complexity with respect to number of IP addresses and is not time dependent (into some extent, it does not matter what time intervals separates any two obtained fingerprints). Moreover, it may run in parallel (although this comes with a possible drawback explained later).

**IPID sampling input.** On the contrary, the IPID sampling process is much more exacting:

- All IPID counter which are to be modeled and compared have to be sampled in parallel in the same time interval.
- The interval between samples (from a particular IP address) have to be larger than several seconds (to avoid rate limiting and out of order replies).
- The interval between samples (from a particular IP address) may not be larger than a couple of minutes. Linear IPID counters typically wraps approximately every 10 minutes. Considering packet loss, it is desired to have several samples from within the wrap interval.
- The more samples taken from each address together with increasing measurement interval, the lower the probability that the counters will be observed as behaving equally by coincidence. However, given a fixed measurement time, more samples from one IP address means a lower number of IP addresses can be sampled within the measurement.
- If all IP addresses would not fit to the time schedule of a single measurement, the set have to be split, but still, every IP address have to be evaluated with respect to every other (all are alias candidates) and so the partial measurements would have to be conducted several times (see naïve splitting algorithm later in this chapter). This will quickly make the method ineffective or even inapplicable.
- The time of one measurement, given a fixed number of IP addresses to probe, is naturally limited by the capacity of the link used for the measurement. The process may be parallelized, but this will introduce

complications with clock synchronization, as pointed out by authors of this alias resolution method [5].

To clarify the constraints assume an example:

Because of the limitation of the link, network, platform or because it is desired to avoid triggering an IDS alarm by intensive measurement traffic, assume a limit was set to send a probe every 20 milliseconds. 5 IPID samples will be requested from each counter's wrap interval (a really low limit, consider packet loss). When assuming the wrap interval is 10 minutes, a probe have to be sent at least every 2 minutes to a particular IP address. In 2 minutes, with 20 millisecond delay, 6000 packets may be sent. If the desired overall number of IPID samples per IP address is set to 30, the measurement will take one hour.

Yet the number of IP addresses for a large AS or a large country may exceed tens of thousands of candidate IP addresses. In the original paper describing IPID Velocity Modeling method [5] authors suggests it is possible to sample 500 000 IP addresses in 17 minutes. However, such measurement sends more than 10 000 probes per second. This may be considered excessive traffic, especially if conducted from non-academic Internet connection.

If there will be no additional information to split this set, the set would have to be split artificially. If the set would be just broken to two halves, and two measurement will be conducted (one for each), then if some IP address from the first half of the set is an alias with some IP address from the second half of the set, such alias will be missed by the method. This is because only IPID counters measured in parallel may be compared.

**Naïve splitting.** Another alternative is to use some naïve splitting algorithm which trades efficiency for completeness. Consider following.

```
INPUT:
M /* the set of candidate IP addresses */
k /* the cardinality of the largest set of candidates that is
possible to measure in parallel */

OUTPUT:
S /* a set containing subsets of M of size c */

   /* assuming integer division */
1. h := (|M| / (k / 2)) + 1 /* nr of half-sized subsets */
2. l := (|M| / h) + 1 /* max nr of elements in 1/2 subsets */
3. Create h subsets (S_n1/2) by moving l elements from M to each
   S_n1/2 (except the last one which may be smaller as it is
   created from what was left in M)
4. Merge each two distinct S_n1/2 to one S_m and add S_m to S
```

**Figure 11:** Naïve splitting algorithm

The algorithm creates subsets of size acceptable for measurement (c), but the larger is the relative difference between M and c, the more subsets will be created. Generally:

$$|S| = \sum_{n=1}^{h} h - n$$

, or the number of non-zero elements in strictly triangular matrix of size h.

**Splitting with fingerprints.** Fortunately, the fingerprinting is efficient enough to avoid the naïve splitting, or at least reduce its input.

A critical role in the success of this type of splitting lies in the fact (or rather call it an assumption) that if some IP address is unresponsive, it may be assumed that it is not an alias of any responsive address. It is sensible to do so, and if it would not be possible, the set of unresponsive IP addresses (that always constitute a relatively large set) would have to be merged with every set of responsive IP addresses. With high probability, it would render the splitting prohibitively inefficient.

Another help is, that if a set of IP addresses is not responsive to a particular probe type (it will be observed during fingerprinting), it naturally makes no sense to sample the IPID of such addresses with this probe type (although for other probe types it is still sensible).

**Used attributes.** After the fingerprints are obtained, and before the IPID sampling, first, the same constraint (AS and countries) is applied to the set of all IP addresses as before the fingerprinting. Next, the fingerprint data are used to split this subset into disjunctive parts.

During this process, the TTL distance from the fingerprinting vantage point is used as one of the splitting attributes. Assuming the country and AS constraint was the same as was used for fingerprinting input, all IP addresses in question were fingerprinted. Hence, it is possible to divide unresponsive addresses from responsive, and for all the responsive addresses the TTL distance from the vantage point is available.

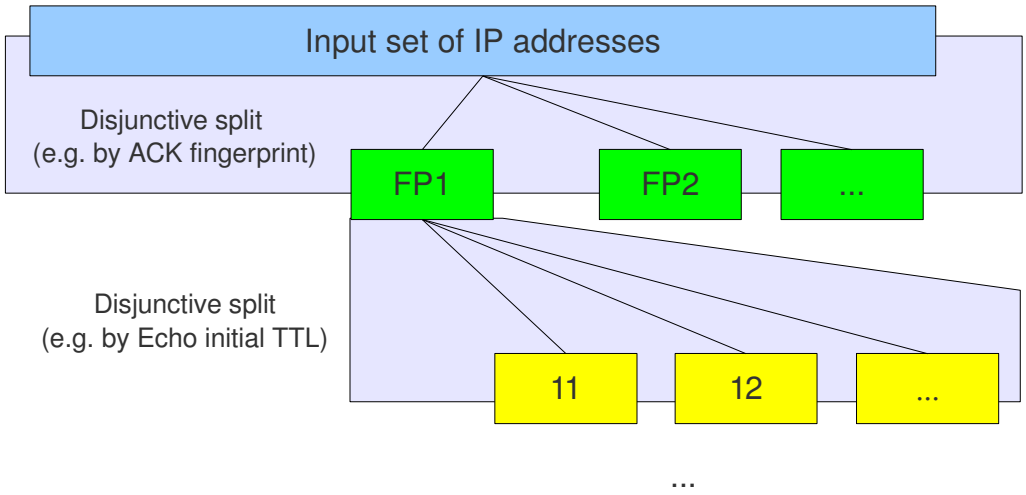



**Figure 12:** Disjunctive splitting.

As the TTL distance, the raw value of TTL field in the reply packet is used (not the hop distance). It is assumed, that for two aliases, this value should differ by no more than 1. It is because the routing in Internet is mainly driven by destination IP address and as alias interfaces will have similar routing tables, the path of the reply packet back to the vantage point should be virtually the same.

On the contrary, the hop distance to two aliases (if available from the traceroute dataset) may be (and probably will be if retrieved from a single vantage point) retrieved from two different traceroutes with different destinations, so the number of hops may vary.

**Splitting with TTL.** The TTL obtained with fingerprinting vantage point may be considered a disjunctive splitting attribute, but the TTL obtained from the traceroute vantage points may not, in general. This is, because some vantage points were probably configured to traceroute different sets of IP addresses. As a result, the distance of a particular IP address may be known to only a subset of vantage points. This causes all the IP addresses, for which the distance is unknown, to be added to each subset of addresses created on the basis of different distance value.

This not only frustrates the splitting algorithm and may render it ineffective, it may actually cause more harm than good, because of the IP address sets with unknown distances being copied several times (as many times as many sets will be created from known distances). Moreover, the TTL may reach any value from 0 to 255 and so theoretically, each vantage point may split the initial set up to 256 subsets. Possibly (and probably) many of these sets will be very small, while still, the addresses with unknown distance (probably a significant number) will be copied to them.

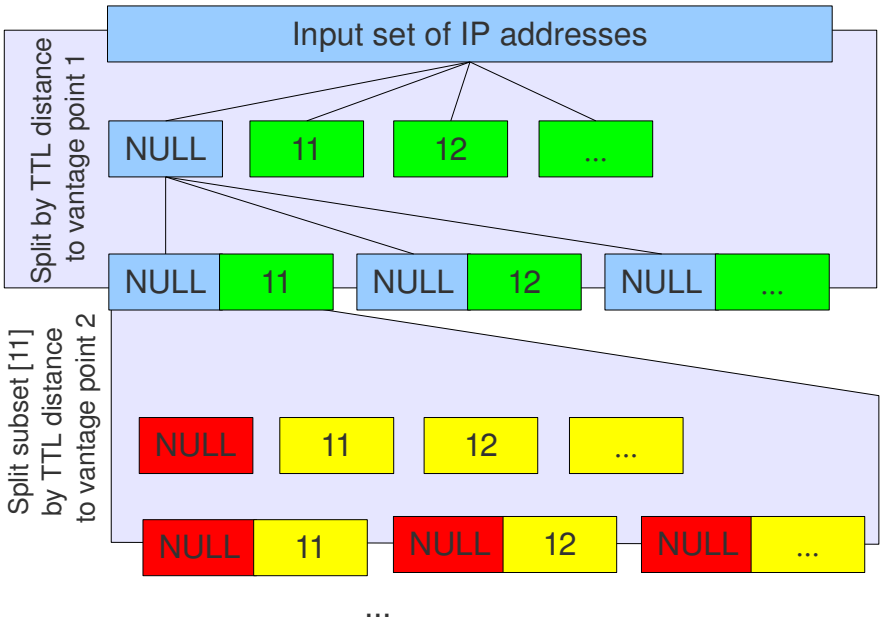



**Figure 13:** Non-disjunctive splitting, causes a rapid increase of overall number of IP addresses (if the subsets with null values are large).

To avoid enormous increase of the overall number of sets and IP addresses, this TTL splitting is the last splitting step (so its input is as small as possible) and the procedure is limited by two parameters:

- Maximum number of vantage points to split by (the sets are split recursively, in each level by another vantage point, so this is the maximum depth of the recursion).
- Maximum fraction the unknown-distance addresses may represent (the fraction vary with respect to particular vantage point).

During the algorithm, the best possible vantage points are dynamically chosen. The vantage point is chosen if the number of unknown-distance IP addresses is minimum among all other vantage points and if this number does not compromise the maximum allowed fraction.

This ensures that the splitting continues only if it can effectively split the set without significantly increasing the overall number of sets and IP addresses.

**Conclusive steps.** At the end of splitting the user is provided with an overview of the number of created subsets and their cardinality. User may decide to adjust splitting properties (if splitting was ineffective) and re-split the set again, and chooses the probe types to use for IPID splitting. Recall that IP addresses which are unresponsive for a particular packet type ends up in a separate subset, hence, such subset will not be sampled if the respective packet type is chosen.

According to cardinality of the largest subset, user can configure the parameters of IPID sampling. If any of the used subset is still too large, the Digester module may be configured to use the naïve splitting algorithm.

Set which was split by naïve algorithm is divided to multiple files. On the contrary, sets too small are merged together to single file to fit the configured cardinality. It is a logical step, as we do not gain anything in sampling files with number of IP addresses smaller than max. cardinality, instead we may loose some aliases (under assumption that the split algorithm incorrectly separated some).

File created by Digester module for the IPID sampling process of Pulsar module is merely the same as for fingerprinting. The one of a few differences is that the IP addresses in the file are not re-probed at the request of captor, but

automatically, according to configured number of samples. The packets sent are almost the same as those sent in fingerprinting process.

### 3.3.4 IPID Velocity Modeling

**Summary.** The principle and algorithm of this method was explained in the chapter regarding related work. Here, mainly its customizations and improvements made in Jardinero tool are described.

**Building IPID counter model.** The samples are obtained from database separately for each measurement and each probe type (it was observed during the development, that samples from replies to UDP packets may use another IPID counter as samples from TCP replies even for the same IP address).

First, the retrieved samples have to be analyzed to find out:

- If there is enough samples for successful modeling.
- What is the wrap (reset) interval of the counter (it is done by searching for a configured number of samples which are in minimal distance and are monotonously increasing).
- The slope of the counter function.
- Whether the counter can be modeled as linear (non-linear “counters” either do not have enough successive samples to even infer the wrap interval, or the slope of produced function is too high - counters wrapping too often).
- Whether the counter is not incrementing the IPID value in big-endian (such counters are transformed to little-endian representation).

After the lists (two: one for big-endian counters and one for little-endian) of linear IPID counters are produced, some non-aliases may be evaluated right away:

- Every counter from big-endian is a non-alias with every little-endian counter.
- Every linear counter is a non-alias with every non-linear counter.

Next the counters from a particular list have to be compared to each other. To avoid the complexity of such comparison ($n^2/2$ comparisons for each list of counters) the counters are sorted by their inferred slopes. Only counters for which slopes do not differ for more than some configured value are compared.

The comparison involves computation of the distance as proposed by authors of the method with these modifications:

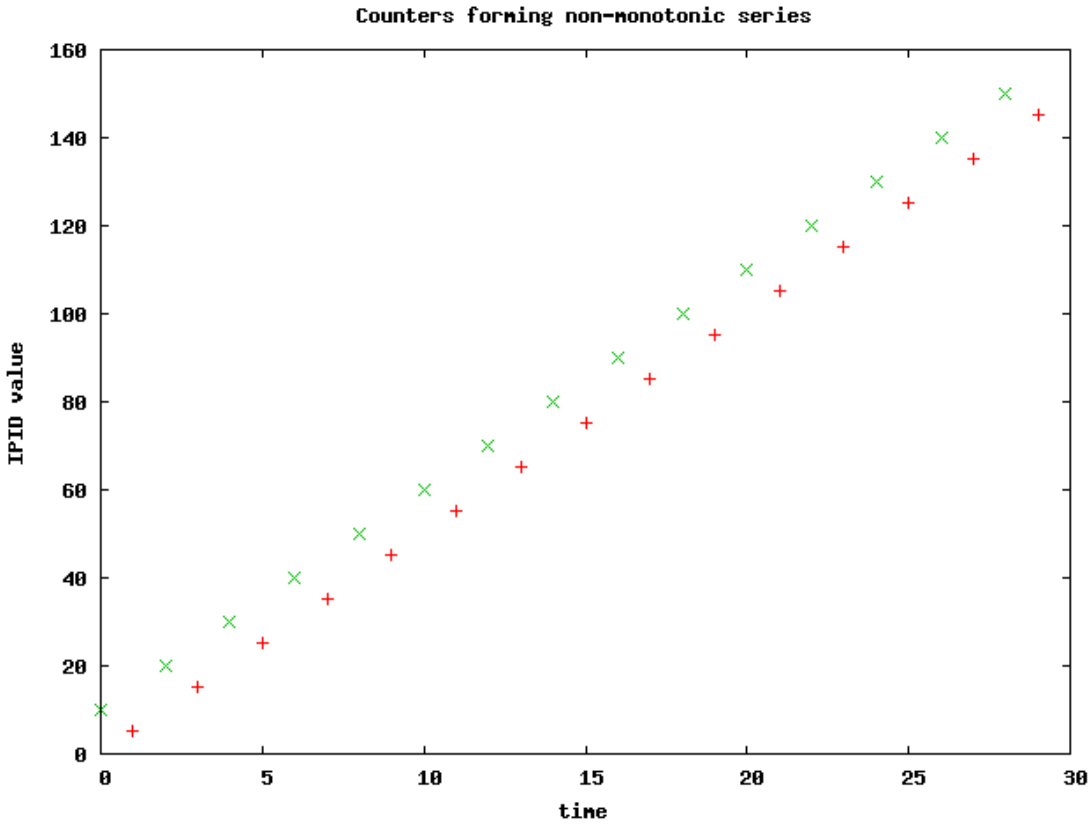



**Figure 14:** An example of two counters which, if merged, do not form monotonic function

- The compared counters are merged together and checked that the resulting model does not wrap. If the samples are collected from one counter, the merged model should not wrap (unless reply packets arrives out of order).

  If any wrap is tolerated, there may only be a small absolute difference in the samples (assuming it was because of out of order replies). For instance it may be 45 010, 45 008 or 65 530, 4 but not 45 010, 50 010.

- The merged counter is also checked for duplicate value as duplicity is not possible in a single counter.

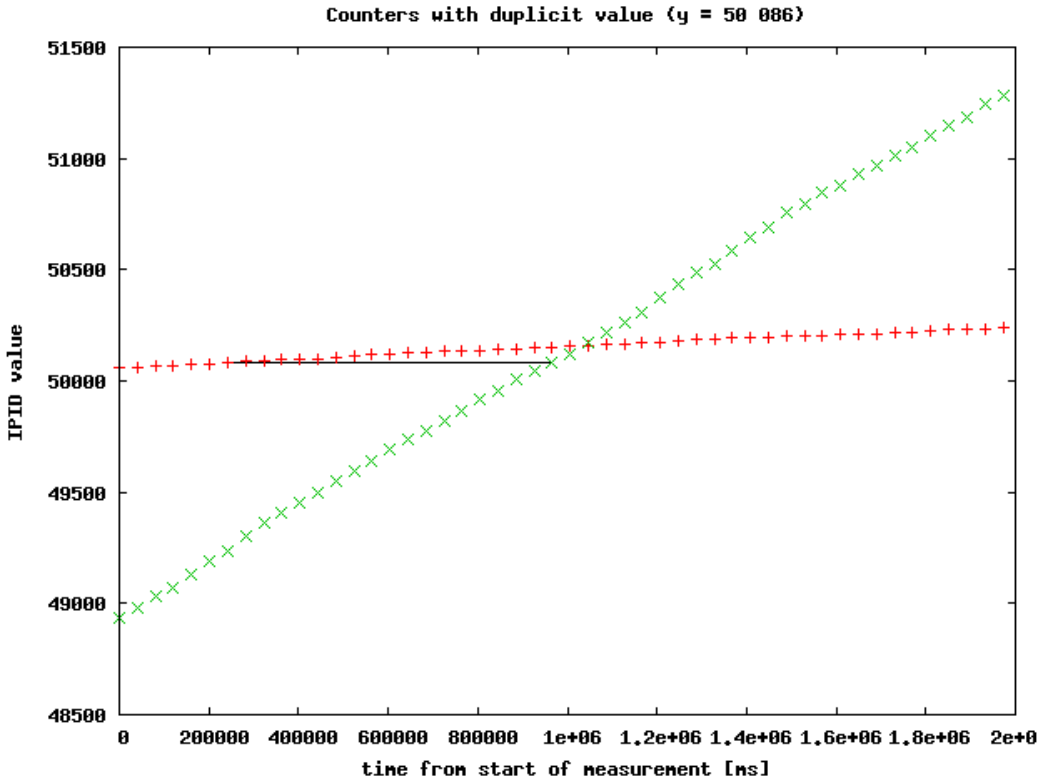



**Figure 15:** Two counters having a duplicate IPID value (connected with horizontal line)

- The distance of elements in head and tail parts are not computed. It is assumed, that not using this less accurate part of distance computation won't have any significant impact, as probably there will be only a couple of samples there – ideally and usually, one in the head and one in the tail.

Finally, if the computed distance is less than specified threshold, the IP addresses are considered aliases. If the distance is above configured threshold, or the slope difference is too high or merged counter is invalidated by described rules, the addresses are evaluated as non-aliases.

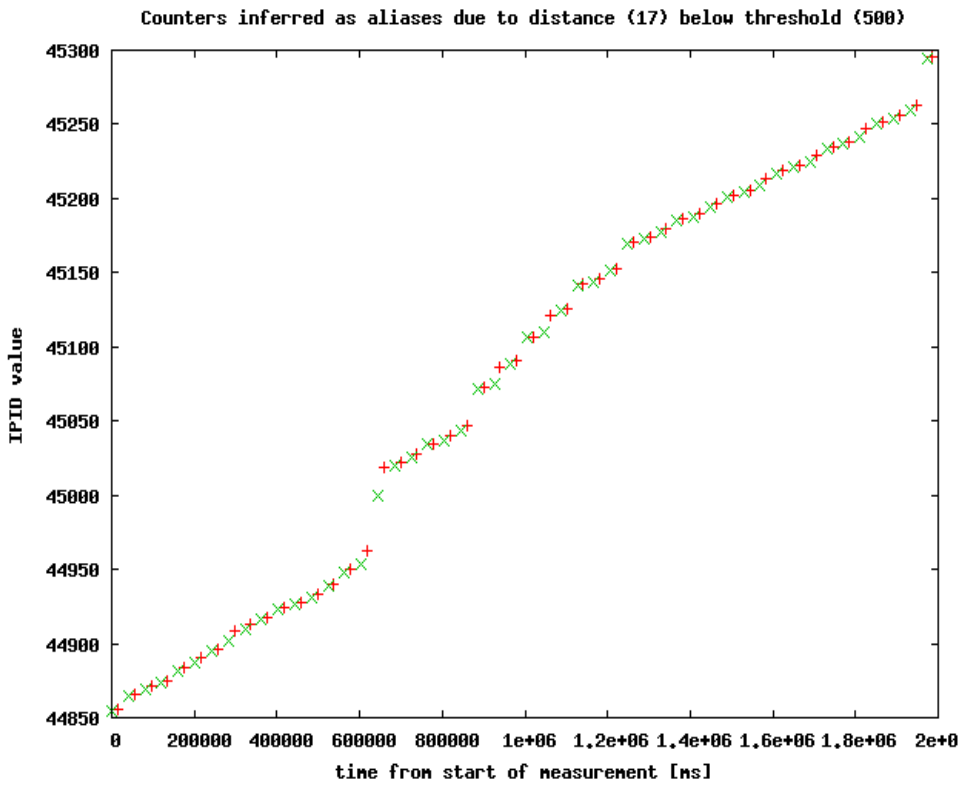



**Figure 16:** Example of alias counter

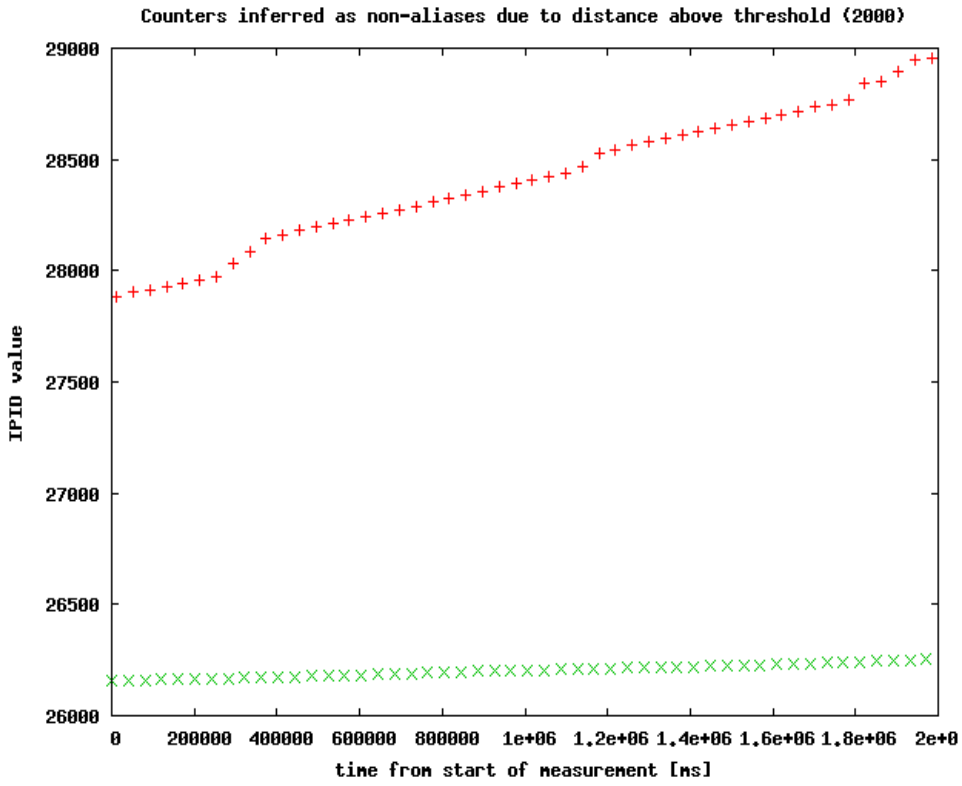



**Figure 17:** Example of non-alias counter

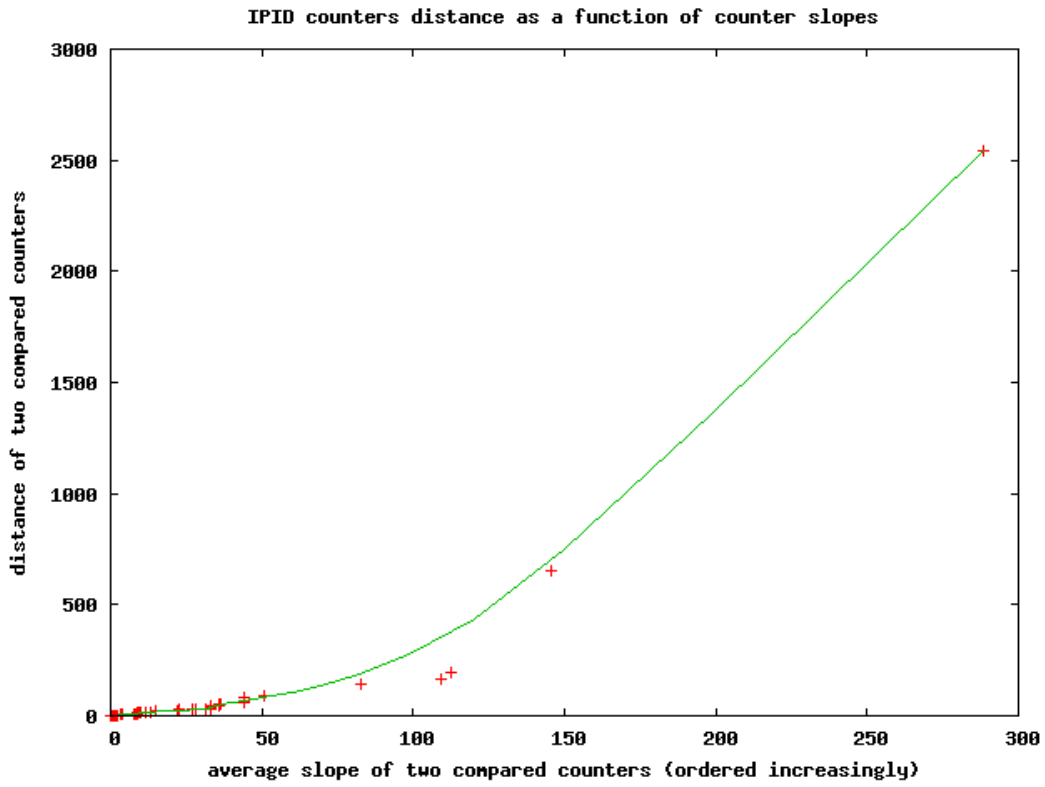



**Figure 18:** Correlation between slope and distance

**Distance and slope.** The method for computing distances has a little quirk. The higher the slope, the higher the computed distance (although the relation in not linear). This is actually eligible, as the higher the slope the higher is the difference in IPID values of successive measured samples. The higher the difference, the less is the confidence that two counters do not overlap accidentally, hence, the higher should be the distance. This causes counters with slope higher than ~300 to have distance (e.g. 520) that is likely to go over the configured alias threshold (e.g. 500). Such counters will not be labeled non-aliases as the non-alias threshold should be far (e.g. 2000), the method simply defer to decide.

This will not happen often, as nearly all linear counters have slopes less than 100.

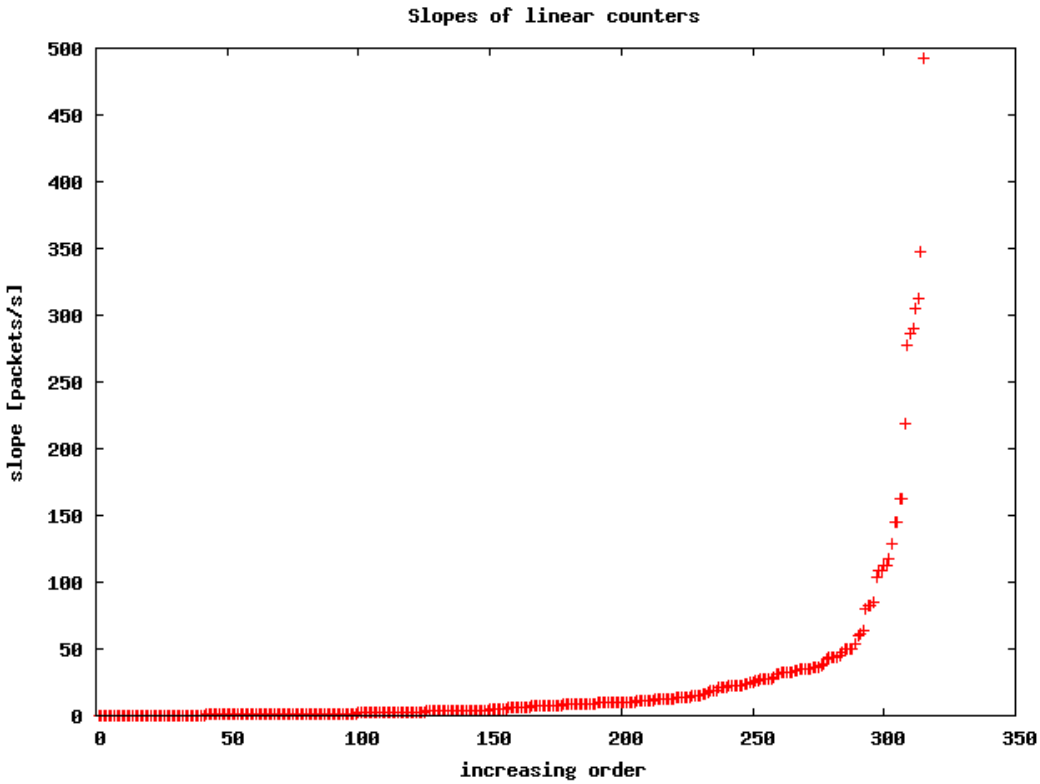



**Figure 19:** Slopes of linear counters from a sample measurement.

### 3.3.5 APAR

**Implementation.** The APAR algorithm was fully implemented on the server side of the database module. This avoids possibility of getting out of memory, and the overhead of copying large data through network.

First, the subnet inferring is executed. With help of data prepared during import of traceroutes, it creates subnets using algorithm described by authors of APAR algorithm [44].

The implementation of APAR itself does not follow the original algorithm fully. First, many of needed data are extracted right from the traceroutes during import procedure (as was described). The improvement introduced by Keys [35] of using a set of 3-hop segment is used.

Also, all already inferred aliases are used during the execution of Common Neighbor rule.

**No-loop condition.** Second, the inefficient implementation of the no-loop rule (which checks if two alias candidates were not seen in a single traceroute) had to be improved. The original algorithm scans all traceroutes for each candidate pair. Keys creates a compressed bitmap for each IP address representing a map of traceroutes in which this IP address occurred. This was not found practical for the use in Jardinero project. Rather a more database-friendly approach was taken. Every traceroute being imported is assigned an ID and every IP address from within the traceroute is mapped to this traceroute with this ID in a dedicated table.

Still it is the most resource consuming part of APAR implementation. During a test run, approximately 60 files containing traceroute dumps were imported. This created more than 90 million records in the ip-to-traceroute mapping table. Even with an b-tree index (which took 2GB of disk space), a single query for two IP addresses takes a couple of seconds.

Fortunately, there are not so many queries. The no-loop check was delayed, and implemented as a post-process procedure. This ensures it is called only for candidates which passed every other constraint. To do this, all aliases found by APAR under authority of aliased neighbors were saved along with these dependent aliases. If a post-process no-loop check finds some of these dependency aliases false, it also discards the dependent alias and then recursively all aliases dependent on discarded alias. This recursive check of dependencies also reads for following two checks (same-fp and non-alias).

**Same-fp condition and non-alias check.** To further reduce the number of no-loop queries, a same-fp check is applied before the no-loop procedure. The same-fp check (as expected) ensures that alias candidates do not differ in their fingerprints.

Next, the non-aliases produced during IPID Velocity Modeling are used to discard possible false aliases.

**Scope of constraints.** The no-loop constraint is applied only for aliases produced by IPID Velocity Modeling and APAR. More confidence is put into aliases produced by Source Address method than into the correctness of traceroute data, so the no-loop rule does not apply for them.

The same-fp constraint is used for all aliases (although if data are fresh it is naturally effective only in finding false positives in aliases produced by APAR, all other aliases already had to pass this condition in case of typical run).

The no-alias check is applied for all aliases, hence, a no-alias from one of the IPID sampling measurement can possibly discard an alias inferred from another measurement. This is logical, as true aliases can not be accidentally measured as non-aliases, while counters of non-aliases may behave as aliases by chance.

**Clustering.** After all aliases were resolved and all constraints applied, the table of aliases contains many alias pairs. Many of these pairs however, describes a single device (alias cluster). If there is alias A-B and alias B-C then there is supposed to be a device with interfaces A,B,C. This clustering is implemented in database and it is the final step of alias resolution.

## 3.4 Evaluation

### 3.4.1 Summary

**Summary.** Various test measurements were conducted several times to help with decisions made during development and to calibrate the tool. Here, for sake of clarity and to support explanation of concepts from previous chapters, only one run is described.

**On the validity of results.** Consider however, that the test executed (despite its large scope) can not be used to infer some invariable behavior of the tool or general success rate of its methods. These properties completely depends on the tested environment. As many times already noted, a method (and this holds for all methods used in Jardinero tool) may yield perfectly accurate and complete results for some autonomous systems or under some conditions, while at the same time may be completely ineffective under other circumstances.

As shown by Willinger *et. al* in [4], the generalization of results of a large-scale network measurement study to whole Internet (even with the use of sophisticated mathematics) may result (and did in past years) in misleading conclusions. It is a very complex task which requires deep understanding of

Internet's network design practices, administration practices, policies used by autonomous systems and mutual relations among them and the rapid and dynamic evolution of Internet. This applies exemplary for IP alias resolution. Changing policies (especially if more restrictive, like dropping of ICMP packets), new technologies (MPLS for instance) or errors and misconfiguration of specious nature (i.e. not decrementing the TTL field) lead to an uncertain error rate. A throughout study of the validity of assumptions used by network measurement studies is beyond the scope and scale of this thesis.

However, this test (and similar studies by other researchers) hopefully provides enough proof that the assumptions on which the tool is based are sensible and the results solid.

### 3.4.2 Import

**Initial import.** IP alias resolution, or at least some parts of it, are dependent on the quality of input data. To test Jardinero tool, publicly available data from iPlane project [15], DIMES project [51] and CAIDA [50] were used. Datasets were described in the chapter explaining import procedures.

iPlane conducts measurements from PlanetLab [57] nodes distributed across globe and targets traceroute effectively, so it can discover as many paths as possible. iPlane generates traceroute dumps daily, in a volume of approximately 200 dump files, each from a distinct PlanetLab node.

For the test, 68 files were imported to database with the import procedures described in previous chapter. All files from vantage points across Europe (as the test was targeted to Europe), and at least one for each other country.

**Results.** Below, various statistics considering the import are presented.

| | |
|---|---|
| Fraction of file imported | 80,0% |
| Size of file | 27 MB |
| Traceroutes in file | 132 000 |
| Traceroutes trimmed | 0,3% |
| Traceroutes trimmed to zero length (dropped) | 2,3% |
| Traceroutes with loop (dropped) | 5,8% |
| Traceroutes with bad IP addresses (dropped) | 5,6% |
| Traceroutes with anonymous routers | 34,2% |
| Hops in file | 2,2 mil |
| Traceroutes imported per second | 24 |
| Hops imported per second | 370 |

**Figure 20:** Rounded average values for 40 of imported files

| | |
|---|---|
| IP addresses | 320 000 |
| 3-hop segments | 2,7 mil |
| IPa – anonym - IPb | 190 000 |
| IPa – anonym - anonym - IPb | 56 000 |
| Relations between AS | 108 000 |
| /24 networks | 275 000 |
| /24 networks inferred nonexistent | 71 000 |
| IP to vantage point distances | 9 mil |
| IP to traceroute records | 96 mil |

**Figure 21:** Status in database after import (rounded values)

### 3.4.3 Fingerprinting

**Input.** The aim of the test was to find IP aliases in GÉANT2 (autonomous system number 20965). GÉANT2 [58] is a network connecting 34 countries through 30 national research and education networks (NRENs). Most of its IP topology can be easily extracted with the Looking Glass tool [59] through which its routers can directly be queried.

To test project's performance on large-scale network measurement, all known peers of the GÉANT2 were included. The result of the autonomous system (20965 + peers) and countries (null) constraint was more than 43 000 IP addresses. IP addresses inferred as hosts were also used, to test how many aliases will be missed if only IP addresses inferred to be routers will be used.

| | |
|---|---|
| Number of IP addresses | 43 103 |
| Number of IP addresses inferred as hosts | 11 308 (26 %) |
| Number of autonomous systems (AS20965 + peers) | 105 |

**Figure 22:** Input after AS and countries constraint

As can be seemed from the following figure, the GÉANT2 network constitute only a marginal part of the input (152 IP addresses). Nevertheless, with the input of only 152 IP addresses it will not be possible to analyze the scalability of the tool.

All prominent autonomous systems in input (except AS174 and AS12965) are Tier-1 networks (AT&T - AS7018, Level3 - AS3356, Qwest – AS209, GBLX – AS3549, etc.). It is probably that if any large AS will be selected for input, if the peers are included, these autonomous systems will be appended.

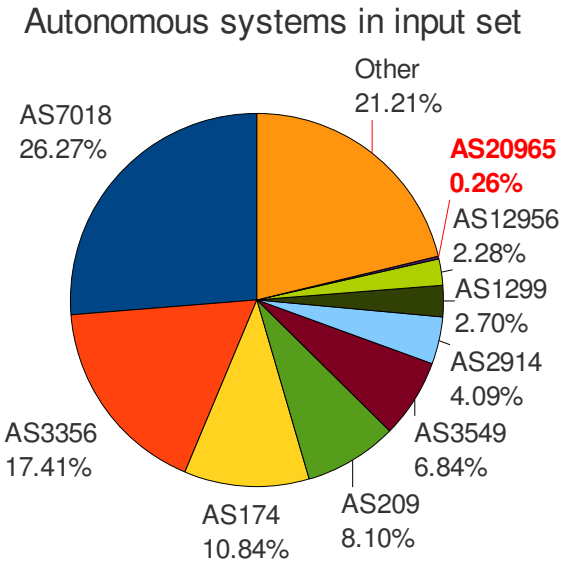



**Figure 23:** Input after AS and countries constraint

**Results.** As a next step, all IP addresses were fingerprinted. Thanks to a high distribution of used PlanetLab nodes, many Source Address aliases were instantly found. Overall statistics of fingerprinting are as follows:

| | |
|---|---|
| Number of unresponsive IP addresses | 8 855 (20%) |
| Number of addresses unresponsive to TCP and UDP | 14 808 (34 %) |
| Number of found IP aliases (Source Address method) | 5 958 |

**Figure 24:** Status after fingerprinting (rounded values)

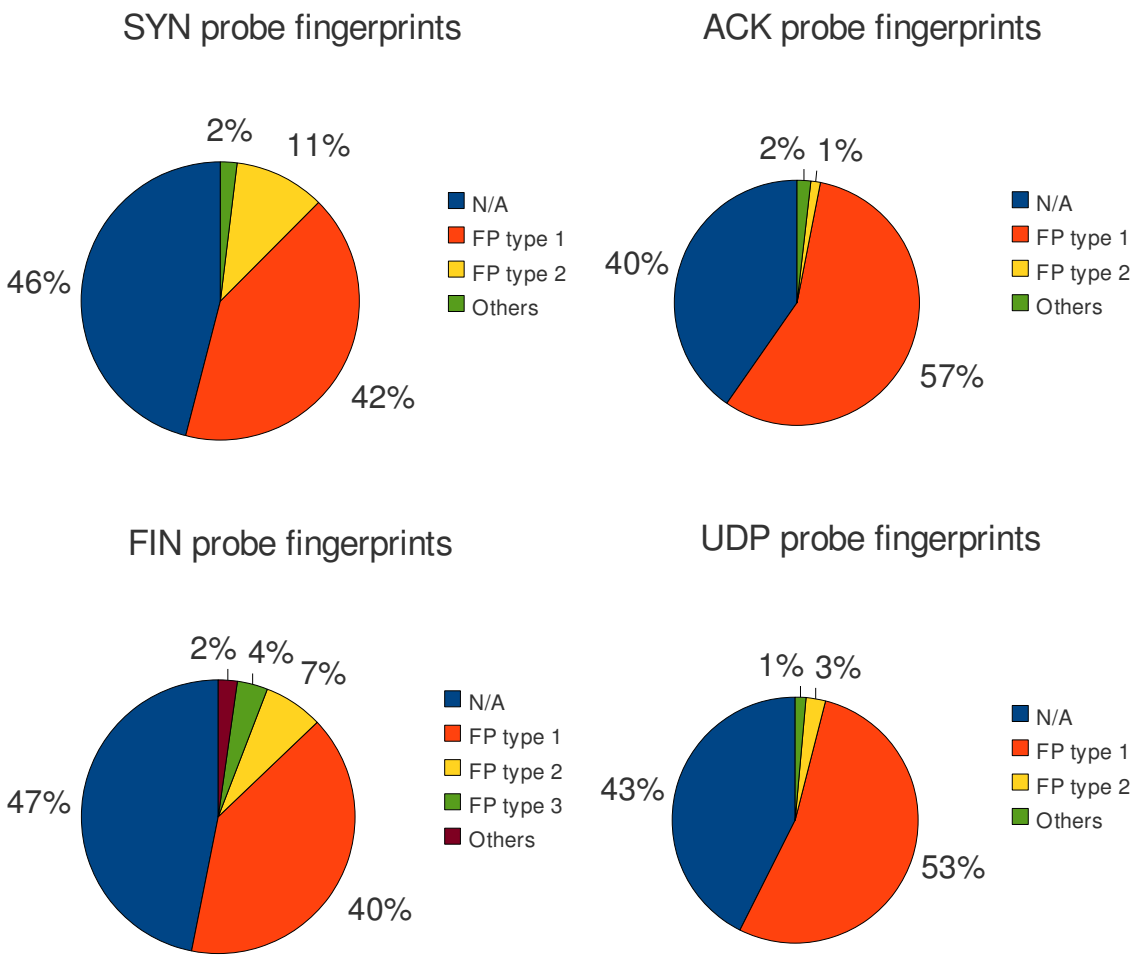



**Figure 25:** Charts representing responses to fingerprinting

For SYN probes, replies mainly differ in setting urgent pointer and in the sequence number handling. For ACK it was urgent pointer, ACK flag and

window size setting, for FIN the differences were in urgent pointer, sequence number and acknowledge number.

ICMP Port Unreachable messages (replies to UDP probes) vary in overall packet length and checksum of UDP header (often zero). All replies to Echo probes were the same (yet are still useful for the value of initial TTL).

Recall that initial TTL as well as TTL (w.r.t. vantage point) are not part of fingerprint, hence a separate figure is presented.

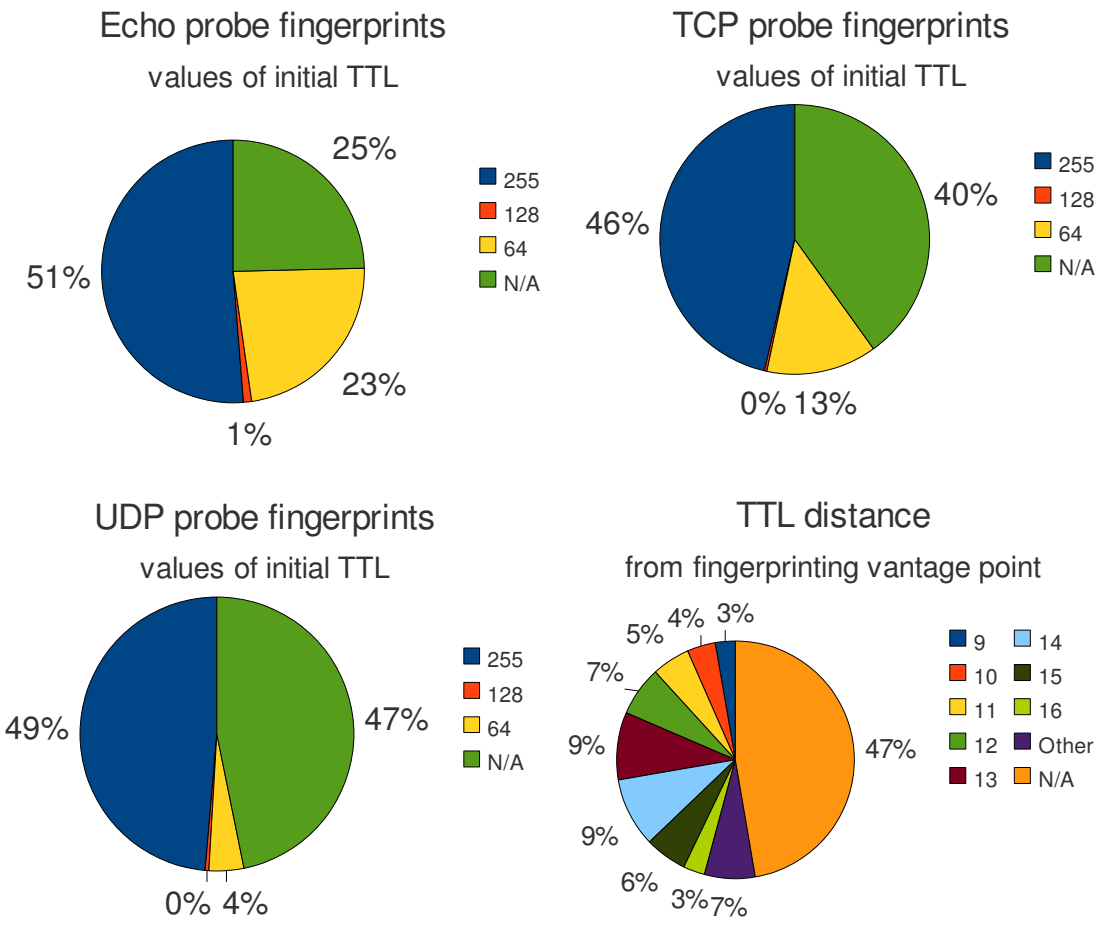



**Figure 26:** Split by initial TTL and TTL distance from fingerprinting vantage point

As expected, the TTL distance distribution has a high peak near values 11-15. This is the reason why it can not be solely used for splitting.

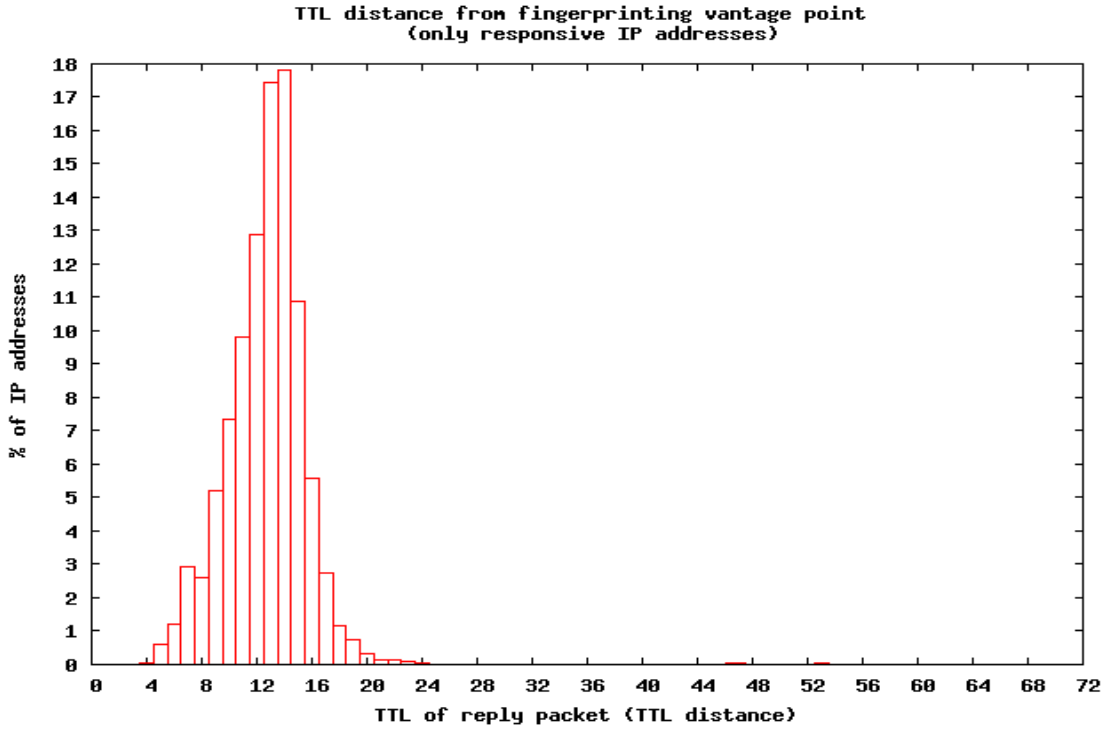



**Figure 27:** Variability of TTL values in reply packets (w.r.t. Fingerprinting vantage point)

**GÉANT2.** Unfortunately, none of the IP addresses belonging to AS20965 was responding to fingerprinting. This excludes this autonomous system from further processing. After inspecting responsiveness of IP addresses from AS11537 (Internet2, the counterpart of GÉANT2 in U.S.A.) the situation was similar. This is frustrating, because it was possible to obtain the IP topology of these autonomous systems and so explicitly verify inferred aliases.

Nevertheless, the test was resumed. After all, at least APAR was able to infer some aliases of GÉANT2. For the rest of discovered aliases, another verification process was used.

### 3.4.4 Splitting

**Input.** As the input to splitting, the same set of IP addresses as for fingerprinting was used. Maximum cardinality of 1400 was chosen.

**Process.** As was explained, the splitting algorithm chooses the next attribute to split by on the fly. Following order was applied during test:

1. Initial TTL of TCP replies
2. Distance from (fingerprinting) vantage point
3. UDP fingerprint
4. FIN fingerprint
5. Initial TTL of Echo replies
6. ACK fingerprint
7. SYN fingerprint
8. Initial TTL of UDP replies
9. Echo fingerprint (as it does not split anything)

The algorithm stops here, claiming no further splitting (by the distance to some vantage point) will be effective.

**Results.** Fortunately, the use of fingerprinting was a success and it helped to solve the splitting problem as the following showy figure illustrates. Although none of the splitting attributes has randomly distributed values, combination of all constraints (fingerprints, initial TTL, TTL distance) was sufficient.

Another advantage of fingerprinting is its ability to disprove false aliases in “same fingerprint” constraint, applied at the end of alias resolution process.

The first three sets (10%, 7% and 7% of overall cardinality) contains IP addresses not responsive to ACK probes and as such are not used for IPID sampling. Next largest set has 1362 IP addresses and so because it is less than the desired cardinality (1400) the naïve splitting algorithm was not used.

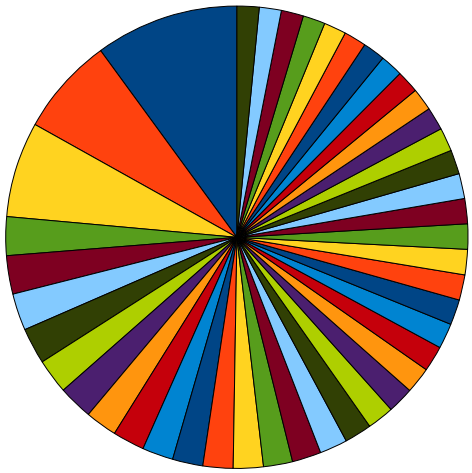


**Figure 28:** Parts represents disjunctive subsets of input. Largest part is 10% wide, next two are 7% and the rest 2% wide.

### 3.4.5 IPID sampling and Velocity Modeling

**Input.** The IPID sampling was only utilized for addresses responsive to fingerprinting. Moreover, only for packets responsive to ACK probes, and only ACK probes were used for sampling. The reason is that ACK probes have higher response rate than other TCP probes.

The UDP probes not only responded poorly (not enough times to collect relevant number of samples), but often the observed function of IPID counter was unrelated to the one observed from TCP packets. This limits the following analytical phase: counters observed by UDP probes may only be compared to each other, but not to the counters observed by TCP probes.

| | |
|---|---|
| Number of samples | 40 |
| Delay between probes | 20ms |
| Measurements (1 for each set/file) | 49 |

**Figure 29:** Configuration

The Echo probes are not suitable for IPID sampling, because the value of IPID in Echo reply is usually non-standard (e.g. it is the same value as in probe).

**Results.** Statistics considering IPID sampling:

| | |
|---|---|
| Number of samplings | 41 307 |
| Number of samples | 1 644 034 |
| Measurement duration | 10 hours |
| Avg. measurement time | 12 min |
| Avg. number of samples | 39,8 |
| Aliases (IPID Velocity Modeling) | 981 |
| Non-aliases (IPID Velocity Modeling) | 2 966 536 |

**Figure 30:** Results

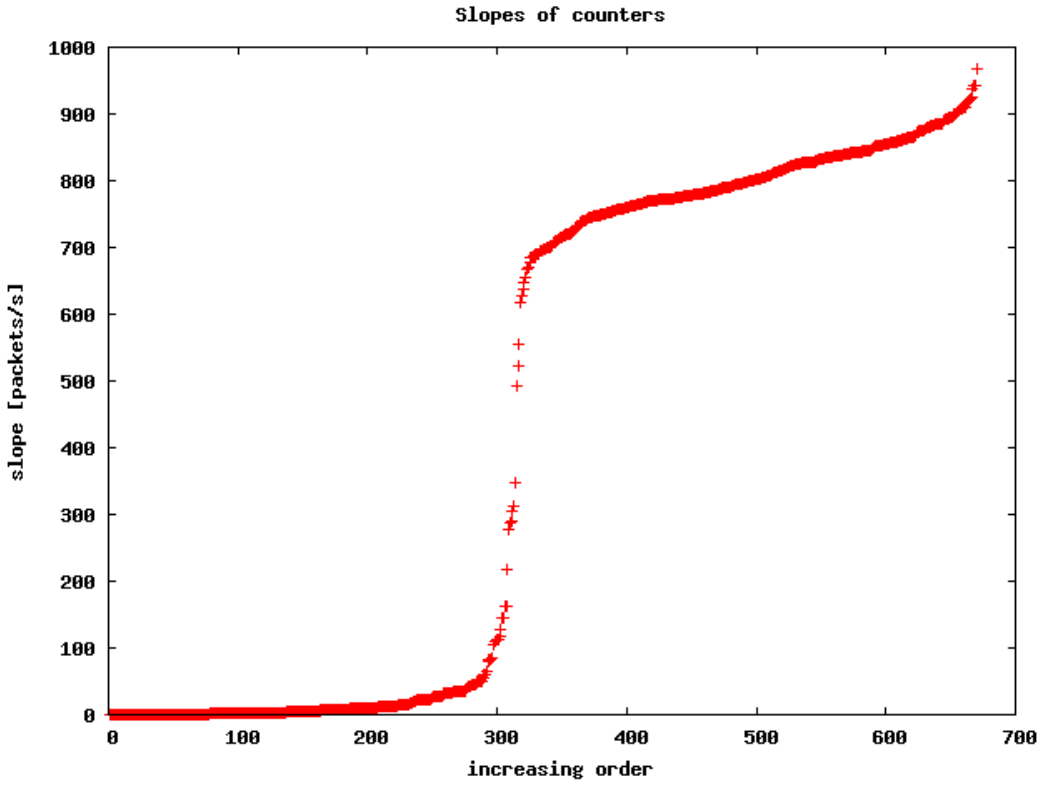



**Figure 31:** Slopes of all counters in one of measurements. Counters with slopes above 500 are considered non-linear.

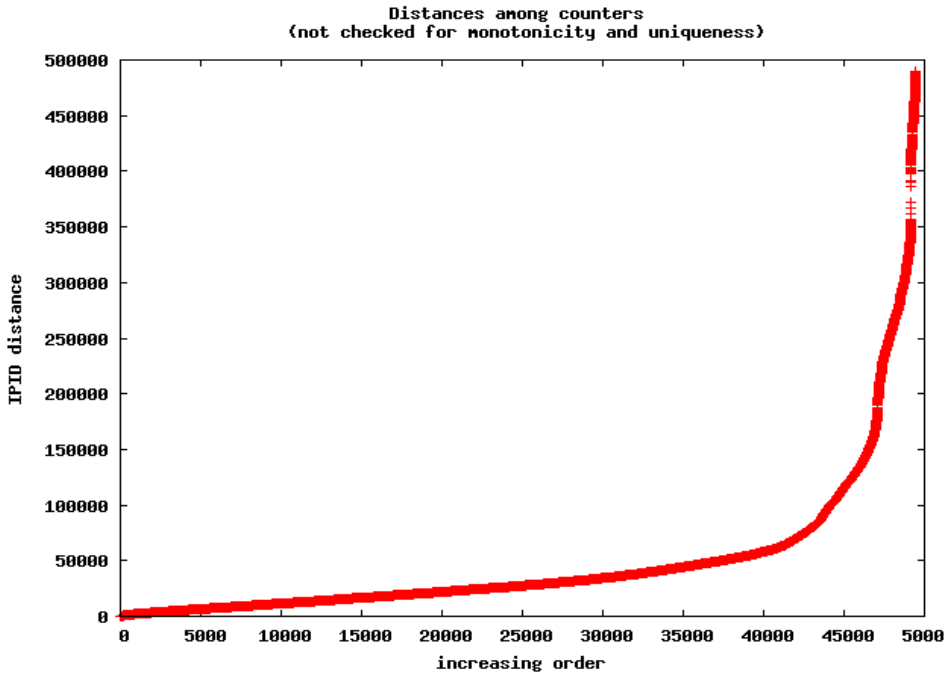



**Figure 32:** Mutual distances of all counters in one of measurements. All distances were specially computed for this figure. Normally, only checked counters with similar slopes are compared.

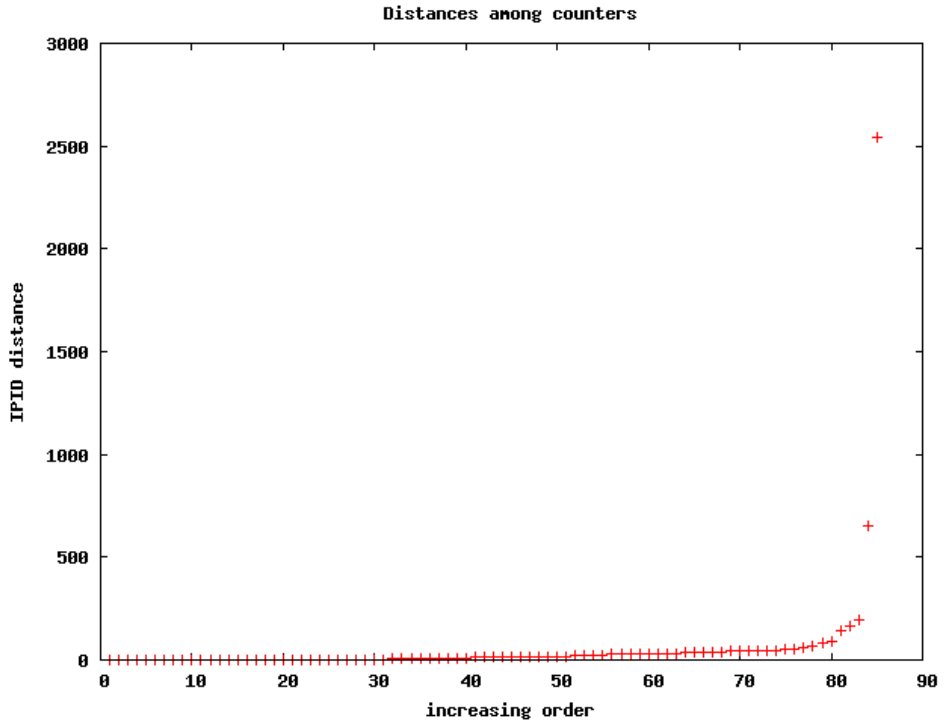



**Figure 33:** Mutual distances of checked counters in one of measurements. It seems that after the counters pass all test and distance can be computed, most of distances are aliases (below alias threshold).

### 3.4.6 APAR

**Input.** The APAR algorithm was, contrary to previous methods, executed on the whole imported dataset. It was the last step, as the algorithm takes advantage of all previous steps. It uses results of fingerprinting to apply fingerprint constraint and already discovered aliases to discover even more aliases during the inference process. Therefore, the APAR's results are first presented for all data, but later also with respect to the autonomous system constraint used for previous active measurements.

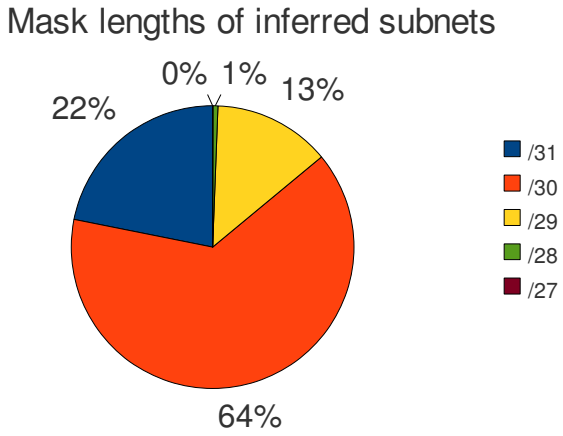



**Figure 34:** Total of 12 486 networks was inferred. As expected, /30 and /31 constitute the majority. Only 4 networks were of size /27.

| | |
|---|---|
| Source Address method | 5 964 |
| IPID Velocity Modeling | 981 |
| APAR first phase | 4 017 |
| APAR second phase | 55 026 |

**Figure 35:** Aliases discovered by APAR and other methods (no constraints applied). Some of Source Address aliases (~10) were discovered during IPID sampling.

| | |
|---|---|
| Source Address method | 5 915 |
| IPID Velocity Modeling | 981 |
| APAR first phase | (-19%) 3 247 |
| APAR second phase | (-26%) 40 289 |

**Figure 36:** After application of “same fingerprint” constraint (15 556 aliases dropped).

| | |
|---|---|
| Source Address method | 5 915 |
| IPID Velocity Modeling | (-1,6%) 965 |
| APAR first phase | 3 247 |
| APAR second phase | (-2) 40 287 |

**Figure 37:** After application of “non-alias” constraint (18 aliases dropped).

| | |
|---|---|
| Source Address method | 5 915 |
| IPID Velocity Modeling | (-2) 963 |
| APAR first phase | (-11) 3 236 |
| APAR second phase | (-17%) 33 409 |

**Figure 38:** After application of “no-loop” constraint (6 890 aliases dropped).

| | |
|---|---|
| Number of discovered aliases | 43 523 |
| Number of alias clusters (distinct routers) | 7 564 |
| Number of distinct IP addresses in alias set | 31 003 |
| Number of “host” IP addresses in previous row | 3 726 (12%) |
| Number of aliases containing “host” IP address | 4 340 (10%) |

**Figure 39:** Results with IP addresses from whole IP dataset

| | |
|---|---|
| Number of discovered aliases | 20 251 |
| Number of alias clusters (distinct routers) | 4 829 |
| Number of distinct IP addresses in alias set | 17 098 |
| Number of “host” IP addresses in previous row | 3 725 (22%) |
| Number of aliases containing “host” IP address | 4 340 (21%) |

**Figure 40:** Final results: results with IP addresses from selected autonomous systems only

Results suggests (last two rows of previous two tables), that APAR does not inferred any alias with IP address labeled (by import procedure) as host. This was expected as APAR can only resolve intermediate nodes in traceroutes.

The routers-only constraint, which can be applied while creating input for fingerprinting or splitting, can (approximately) reduce the input by 26% while this causes a loss of 21% of aliases.

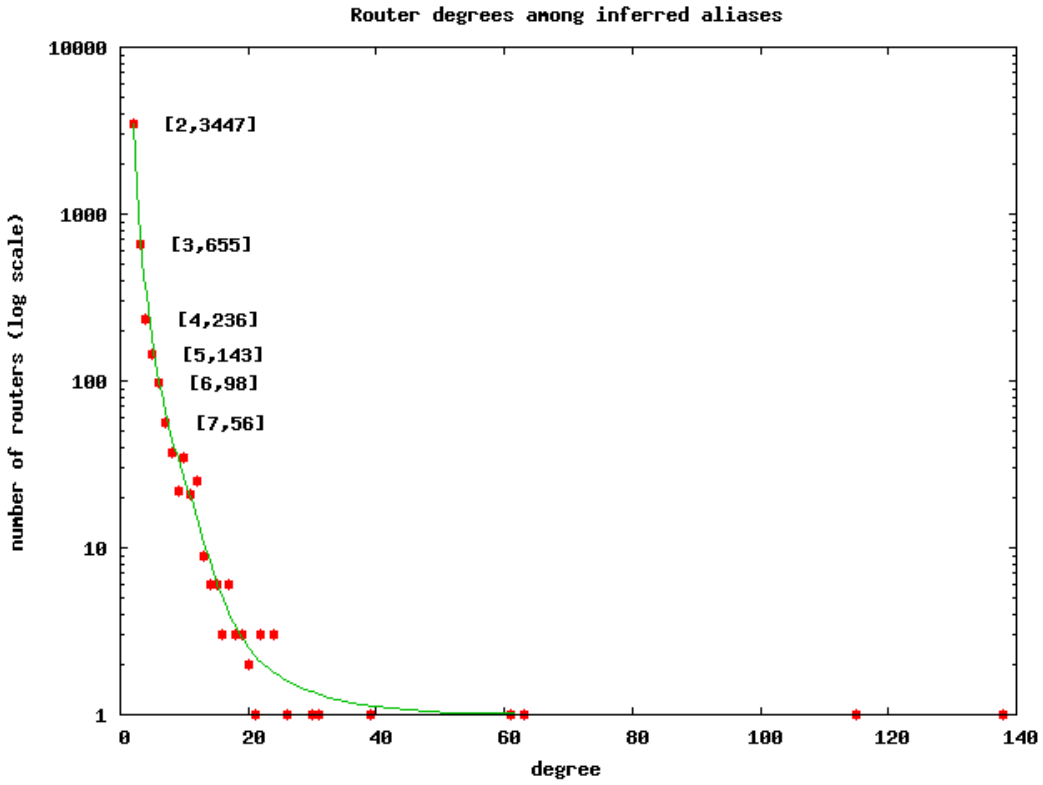



**Figure 41:** Router (alias cluster) degree distribution. One measurement was discarded (2971,1) as it was clearly false.

**Validation.** There are two basic metrics for results of IP alias resolution. Accuracy and completeness. Accuracy describes what fraction of results are false aliases or false non-aliases. Completeness describes what fraction of existing aliases the method discovers.

As mentioned, it is generally not possible to obtain a set of true existing aliases, except for autonomous systems like GÉANT2 with public IP network topology (or part of it), because network topologies are considered confidential.

For available topologies, the results are validated in an exact manner. For other autonomous systems, following heuristics was used:

- Hostnames of all IP addresses were obtained (if available).
- Because AS administrators usually use some naming convention for hostnames, alias interfaces tends to have similar hostnames (this was exploited by Spring *et al.* in [14]). For instance:

  g5-1.inr-250-reccev.Berkeley.EDU

  g5-2.inr-250-reccev.Berkeley.EDU
- All aliases and non-aliases are tested as follows:
  - The hostnames are split by the dot character (.) and then by the dash character (-).
  - A fraction of matching parts is computed. For instance, the hostnames from the example have 7 parts, where one of them is distinct. This gives 85% match.
- Hypothetically, aliases should be giving a high value in average.

This heuristics is unreliable, as all routers in the same autonomous system will tend to have high match value. On the other hand, after visually inspecting the results, some interfaces are likely aliases (the structure of hostname suggests) but the match value is 0 (for instance because names are shifted by one position). Good results will not proof the aliases are accurate, but at least may raise the confidence in the results, if aliases and non-aliases will significantly differ.

**GÉANT2.** It was possible to query 18 routers and obtain the configuration of their interfaces. Among discovered aliases, there were 5 routers inferred. Every inferred interface of those routers was existent and accurate.

| | Budapest | London | Tallinn | Vilnius | Riga |
|---|---|---|---|---|---|
| Interfaces discovered | 3 | 7 | 2 | 2 | 2 |
| Interfaces | 19 | 27 | 9 | 9 | 9 |

**Figure 42:** Discovered GÉANT2 routers

However, not all interfaces were discovered. Because only APAR was involved, this may have two reasons: either the interfaces were not contained within traceroute collection, or the interfaces are not configured to route public traffic. This is very probable at least for some of them. From the interface description, it was sometimes clear it is a backup interface or some private/experimental VLAN.

**Hostname heuristics.** Statistics for alias set and non-alias set computed with the hostname heuristics follows. 12 834 aliases were comparable (both hostnames were available). The same number of non-aliases was randomly selected from comparable non-aliases.

| | |
|---|---|
| Number of comparisons | 12 834 |
| Average match between aliases | 46,82 |
| Average match between non-aliases | 7,92 |

**Figure 43:** Results for hostname heuristics.

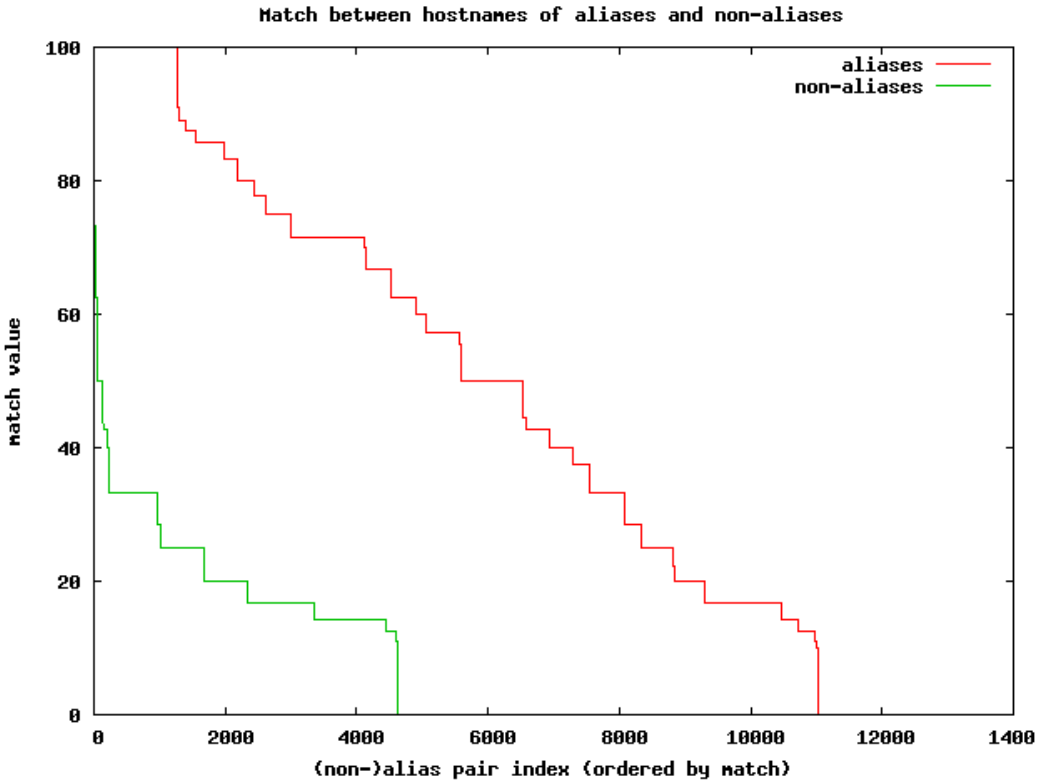



**Figure 44:** Results for hostname heuristics.

**Time cost.** Following table sums the time it took to execute individual steps of alias resolution during the test run. Some of values shown could be lowered by setting more aggressive approaches (or, of course, taking smaller input). Some will be hopefully improved by future optimizations of the software. Overall, the values are far from what will be desired. Consider however, that some of the applied procedures are very demanding and for such a large dataset, it was not even possible to run them in their original implementation.

| | |
|---|---|
| Import routines | 80 h |
| Applying AS and countries constraint | 17 min |
| Fingerprinting (unresponsive IPs re-probed 3 times, all 5 probe types, 100ms delay between packets) | 11 h |
| Import of fingerprinting results | 15 min |
| Computing distances from all vantage points to all IPs | 22 min |
| Splitting (without recreating the AS and countries constraint) | 3 min |
| IPID sampling (20ms delay between packets) | 10 h |
| Import of IPID sampling results | 30 min |
| IPID Velocity Modeling | 1 h |
| Same fingerprint and non-alias constraints | 1 min |
| No-loop constraint | 7 h |
| Creating alias clusters | 18 s |

**Figure 45:** Rounded time-costs of alias resolution steps.

**Obtrusiveness.** During and after the measurements, no abuse reports or complaints from the side of ISP were received.

# 4 Conclusion

**Problem.** The aim of this thesis was to contribute to the solution of IP alias resolution problem. IP alias resolution is a process of inferring which IP addresses are addresses of interfaces of a particular router. This is a crucial part of network topology measurements (especially if executed from multiple vantage points) were the traceroute tool is used.

**Contribution.** First, most contemporary methods of IP alias resolution were thoroughly analyzed. Next, a hybrid approach with several improvements was suggested and implemented in the Jardinero tool.

Virtually every aspect of the tool was studied and evaluated in the subsequent test.

**Result.** The results of the evaluation suggests the IP alias resolution process of the Jardinero tool is both accurate (as much as can be inferred from available data) and complete (as much as possible with given input and network environment settings). Moreover, the study shows many interesting properties of tested networks (responsiveness, router diversity, etc.).

**Future work.** Continuous work will focus on optimization of some of the involved procedures (e.g. tuning the database functions) as well as implementing new improvements (e.g. using TTL-limited probes). The Jardinero tool is freely available to the community to use or contribute.

See https://gforge.cythres.cz/gf/project/jardinero/ for latest release or more details.

# Index of objects

# Attached electronic data

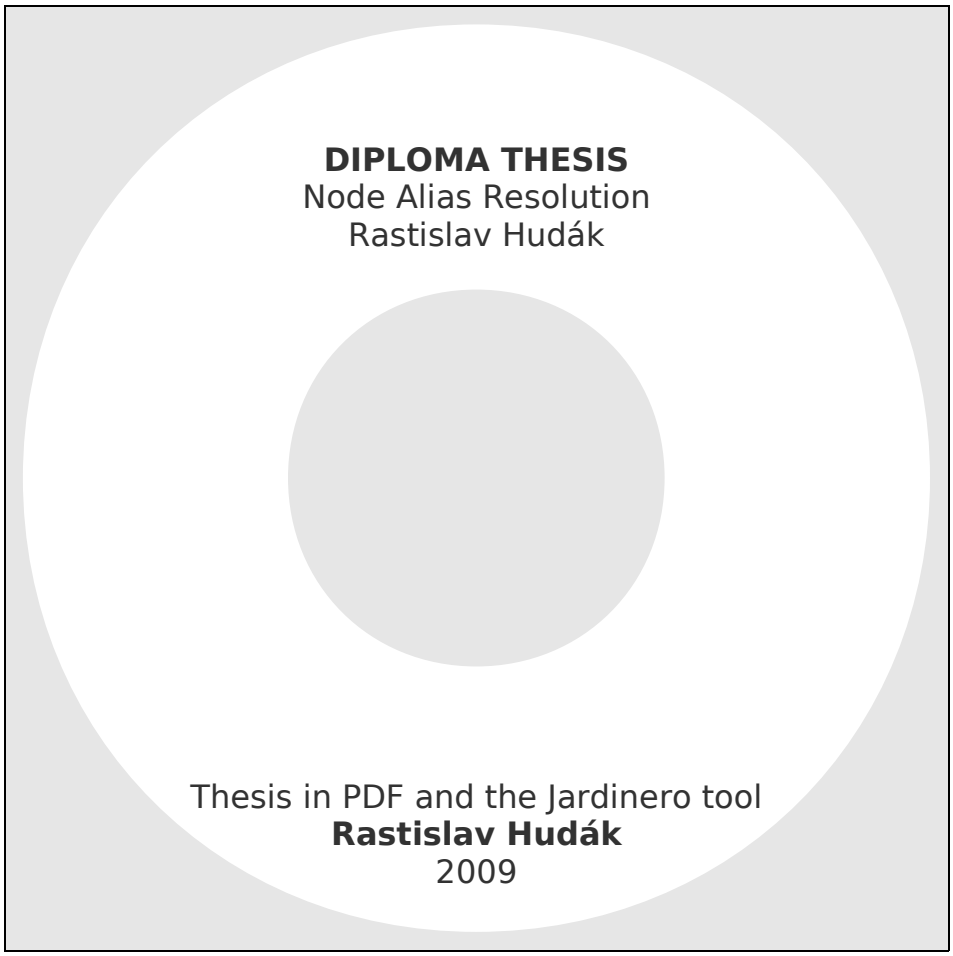
## Description of attached files and folders[11]

| | |
|---|---|
| CONTENT.txt | Describes content of the DVD |
| Thesis | |
| DT_Hudak_NAR.pdf | PDF form of this thesis |
| Jardinero | Directory containing the Jardinero tool |
| README.txt | HOWTO for running the Jardinero tool |
| Dist | Compiled distribution |
| Live | Contains a preprocessed distribution with results of test described in this thesis |
| Project | Exported SVN repository |

[11]You might find additional auxiliary files for download at this thesis' web location.

# Accessing the attached electronic data

The paper version of this thesis should contain an envelope with an optical data medium (CD or DVD) here.



# 5 Bibliography

[1] R. Teixeira, K. Marzullo, S. Savage, and G.M. Voelker, In search of path diversity in ISP networks, Internet Measurement Conference, 2003, pp. 313-318.

[2] M.H. Gunes and K. Sarac, Importance of IP alias resolution in sampling Internet topologies, IEEE Global Internet Symposium, 2007, pp. 19-24.

[3] M.H. Gunes, N.S. Nielsen, and K. Sarac, Impact of Alias Resolution on Traceroute-Based Sample Network Topologies, Lecture Notes in Computer Science, vol. 4427, 2007, p. 260.

[4] W. Willinger, D. Alderson, J.C. Doyle, Mathematics and Internet: A Source of Enormous Confusion and Great Potential, Notices of the American Mathematical Society, vol. 56, 2009, pp. 596-599.

[5] A. Bender, R. Sherwood, and N. Spring, Fixing ally's growing pains with velocity modeling, ACM New York, NY, USA, 2008, pp. 337-342.

[6] R. Braden, Requirements for Internet Hosts - Communication Layers, RFC 1122, IETF, 1989.

[7] P. Ferguson, D. Senie, Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing, RFC 2827, IETF, 2000.

[8] H. Burch, Eliminating Machine Duplicity in Traceroute-based Internet Topology Measurements, CMU Technical Report, 2002.

[9] J.J. Pansiot and D. Grad, On Routes and Multicast Trees in the lnternet, Université Louis Pasteur, 1998.

[10] F. Baker, Requirements for IP Version 4 Routers, RFC 1812, IETF, 1995.

[11] P. Barford, A. Bestavros, J. Byers, and M. Crovella, On the marginal utility of network topology measurements, ACM New York, 2001, pp. 5-17.

[12] CAIDA, skitter tool, http://www.caida.org/tools/measurement/skitter/, 1998.

[13] M. Luckie, Y. Hyun, and B. Huffaker, Traceroute probe method and forward IP path inference, ACM New York, 2008, pp. 311-324.

[14] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, Measuring ISP topologies with Rocketfuel, IEEE/ACM Transactions On Networking, vol. 12, 2004, pp. 2-16.

[15] H.V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, iPlane: An information plane for distributed services, Proceedings of the 7th symposium on Operating systems design and implementation, 2006, pp. 367-380.

[16] H. Tangmunarunkit, R. Govidan, and S. Shenker, Internet topology: discovery and policy impact, Oxford University Press, 2003.

[17] V. Paxson, End-to-end routing behavior in the Internet, ACM SIGCOMM Computer Communication Review, vol. 36, 2006, pp. 41-56.

[18] A. Lakhina, J.W. Byers, M. Crovella, and P. Xie, Sampling biases in IP topology measurements, In IEEE INFOCOM, 2003, pp. 332-341.

[19] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, Avoiding traceroute anomalies with Paris traceroute, ACM New York, 2006, pp. 153-158.

[20] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, Topology inference in the presence of anonymous routers, In IEEE INFOCOM, 2003, pp. 353-363.

[21] S. Bilir, G.P.I.C. Science, and U.O.T.A. Dallas, Intersection characteristics of end-to-end Internet paths and trees, University of Texas at Dallas, 2005.

[22] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, Efficient algorithms for large-scale topology discovery, ACM New York, 2005, pp. 327-338.

[23] A. Zeitoun and S. Jamin, Rapid exploration of internet live address space using optimal discovery path, In Proceedings of Global Communications Conference, 2003.

[24] Information Sciences Institute, University of Southern California, RFC 791, IETF, 1981.

[25] National Laboratory for Applied Network Research, IP Measurement Protocol (IPMP), http://www.nlanr.net/ActMon/IPMP/ipmp.html, 1998.

[26] A.C. Snoeren, Hash-based IP traceback, ACM New York, 2001, pp. 3-14.

[27] W. Chen, Y. Huang, B. Ribeiro, K. Suh, H. Zhang, E. de Souza, J. Kurose, and D. Towsley, Exploiting the ipid field to infer network path and end-system characteristics. In Proceedings of the Passive and Active Measurement Workshop, 2005.

[28] S.M. Bellovin, A technique for counting NATted hosts, ACM New York, 2002, pp. 267-272.

[29] G. Lyon, TCP Idle Scan (-sI), Nmap Network Scanning, Nmap Project, 2009, pp. 117-124.

[30] M. Zalewski, p0f v2, http://lcamtuf.coredump.cx/p0f.shtml, 2009.

[31] G. Lyon, TCP/IP Fingerprinting Methods Supported by Nmap, Nmap Network Scanning, Nmap Project, 2009, pp. 176-189.

[32] N. Spring, M. Dontcheva, M. Rodrig, and D. Wetherall, How to resolve IP aliases, Tech. Report 04-05-04, Washington Univ. Computer Sci., 2004, pp. 04-05.

[33] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, Address Allocation for Private Internets, RFC 1918, IETF, 1996.

[34] C. Labovitz and A. Ahuja, Shining Light on Dark Internet Address Space, NANOG 23, 2001.

[35] K. Keys, IP Alias Resolution Techniques, version 1.1, http://www.caida.org/publications/papers/2008/alias_resolution_techreport/, CAIDA, 2009.

[36] N. Feamster, D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, Measuring the Effects of Internet Path Faults on Reactive Routing, In Proceedings of ACM SIGMETRICS, 2003, pp. 126-137.

[37] A. Botta, W. de Donato, A. Pescapè, G. Ventre, Discovering Topologies at Router Level: Part II, Globecom 2007, 2007.

[38] S. García-Jiménez, E. Magaña, D. Morató y M. Izal, Techniques for Better Alias Resolution in Internet Topology Discovery, IM 2009 mini conference, 2009.

[39] B. Huffaker, CAIDA's Topology Updates and Analysis, CAIDA/WIDE/CASFI Workshop, 2009.

[40] R. Sherwood and N. Spring, Touring the Internet in a TCP Sidecar, ACM New York, 2006, pp. 339-344.

[41] R. Sherwood, A. Bender, and N. Spring, Discarte: a Disjunctive Internet Cartographer, ACM SIGCOMM Computer Communication Review, vol. 38, issue 4, 2008, pp. 304-314.

[42] M. H. Gunes and K. Sarac, Analytical IP Alias Resolution, International Conference on Communications, 2006.

[43] M. H. Gunes and K. Sarac, Resolving IP aliases in building traceroute-based Internet maps, University of Texas at Dallas, Tech. Rep, 2006.

[44] M. H. Gunes and K. Sarac, Inferring Subnets in Router-level Topology Collection Studies, ACM New York, 2007, pp. 203-208.

[45] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, J. Postel, Internet Registry IP Allocation Guidelines, RFC 2050, IETF, 1996.

[46] A. Retana, R. White, V. Fuller, D. McPherson, Using 31-Bit Prefixes on IPv4 Point-to-Point Links, RFC 3021, IETF, 2000.

[47] J. Mogul, J. Postel, Internet Standard Subnetting Procedure, RFC 950, IETF, 1985.

[48] R. Siamwalla, R. Sharma, S. Keshav, Discovering Internet Topology, Cornell Network Research Group, Cornell University, Ithaca, Submitted to IEEE INFOCOM 99, 1999.

[49] CAIDA, The CAIDA AS Relationships Dataset, 6.-7.2009, http://www.caida.org/data/active/as-relationships/, 2009.

[50] Y. Hyun, B. Huffaker, D. Andersen, E. Aben, M. Luckie, kc claffy, C. Shannon, The IPv4 Routed /24 AS Links Dataset, 6.-7.2009, http://www.caida.org/data/active/ipv4_routed_topology_aslinks_dataset.xml, 2009.

[51] Y. Shavitt and E. Shir, DIMES - Letting the Internet Measure Itself, ACM SIGCOMM Computer Communication Review, vol. 35, issue 5, 2005, pp. 71-74.

[52] CAIDA, The Network to AS Mapping Dataset, 6.-7.2009, http://data.caida.org/datasets/routing/routeviews-prefix2as/, 2009.

[53] A. P. Jacobsen, The DNSBL countries.nerd.dk., http://countries.nerd.dk/more.html, 2009.

[54] S. Deering, Host extensions for IP multicasting, RFC 1112, IETF, 1989.

[55] Honeynet Project, Know Your Enemy: Passive Fingerprinting, http://www.honeynet.org/papers/finger, 2002.

[56] J. Postel, Internet Control Message Protocol, RFC 792, IETF, 1981.

[57] PlanetLab Community, PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services, http://www.planet-lab.org/, 2009.

[58] GÉANT2 TransEurasia Information Network - the Next Generation, http://www.geant2.net/, 2009.

[59] GÉANT2 Aujaar - The revamped Stats Portal, http://stats.geant2.net/, 2009.